Identificación de imágenes por análisis de texturas utilizando aprendizaje automático

Trabajo de fin de grado

Nicolás Alcaine Camilli
Alejandro Rodríguez Chacón

Departamento de Arquitectura de Computadores y Automática
Facultad de Informática
Universidad Complutense de Madrid

Curso académico 2018/2019

Identificación de imágenes por análisis de texturas utilizando aprendizaje automático

Memoria presentada para el trabajo de fin de grado de
Ingeniería Informática

Dirigida por el Doctor
José Jaime Ruz Ortiz

Departamento de Arquitectura de Computadores y Automática
Facultad de Informática
Universidad Complutense de Madrid

Curso académico 2018/2019

Agradecimientos

Queremos agradecer los ánimos, la ayuda y el conocimiento recibidos a una gran cantidad de personas. Sin ellas, este trabajo no habría sido posible.

En primer lugar, queremos referirnos a nuestro tutor, José Jaime Ruz Ortiz, por permitirnos realizar este TFG, por su buena fe, por su paciencia y por ser una fuente de estímulo y conocimiento inestimable.

También queremos dar las gracias a nuestros familiares, compañeros en la facultad y amigos. Ellos han sido un sostén frente a la adversidad, pues siempre nos han transmitido sus mejores deseos y nos han ofrecido su ayuda incondicional.

No nos olvidamos tampoco de todos los profesores, que, durante la carrera, gracias a su labor docente, nos han preparado para afrontar el futuro con la mayor de las seguridades.

Por último, debemos agradecer a todos aquellos autores, pasados o presentes, que, bien a través de la publicación de libros o artículos, o bien mediante el desarrollo de la técnica, han creado el sedimento que hace posible, no solo este trabajo, sino cualquier otro trabajo, actual o futuro.

Resumen

En la actualidad, el reconocimiento de objetos en imágenes a través del procesamiento digital y el aprendizaje automático es un campo que se encuentra en claro auge dentro del mundo moderno, esto es así hasta el punto de que la utilización de las tecnologías derivadas de esta área de investigación han generado múltiples dilemas éticos en la sociedad actual. Para poder aseverar esta afirmación no tenemos más que rememorar las distintas polémicas causadas, recientemente, por los sistemas de reconocimiento facial.

Dentro del procesamiento digital por aprendizaje automático encontramos el análisis de textura como una de las variantes más prominentes en el organigrama de este extenso campo. Y precisamente este será el tema fundamental de nuestro proyecto, el cual desarrollaremos y profundizaremos a lo largo de estas páginas.

Por ende, nuestro objetivo para este proyecto será implementar un entorno operativo que facilite el desarrollo de diferentes aplicaciones de clasificación de texturas y que permita ensayos con diferentes parámetros y configuraciones. Además, aplicaremos el sistema, a modo de ensayo práctico, sobre el problema de identificación de la neumonía a partir de radiografías con rayos X.

Para llevar a cabo esto, el proyecto se dividirá en varias partes diferenciadas:

- Investigación y estudio previo.
- Desarrollo de la aplicación en tres módulos funcionales.
- Testeo para la búsqueda de posibles errores de implementación y búsqueda de la configuración óptima para el problema de la neumonía.
- Resultados y conclusiones.

Abstract

Nowadays, the recognition of objects in images through digital processing and machine learning is a field that is clearly booming in the modern world, this is so to the point that the use of technologies derived from this area of research has generated multiple ethical dilemmas in today's society. To be able to assert this affirmation we have only to remember the different controversies caused, recently, by the facial recognition systems.

Within the digital processing by machine learning we find texture analysis as one of the most prominent variants in the organization chart of this extensive field. And precisely this will be the fundamental theme of our project, which we will develop and deepen throughout this pages.

Therefore, our goal for this project will be to implement an operating environment that facilitates the development of different textural classification applications and allows testing with different parameters and configurations. In addition, we will apply the system, as a practical test, on the problem of identifying pneumonia from x-rays.

To carry out this, the project will be divided into several differentiated parts:

- Research and previous study.
- Development of the application in three functional modules.
- Testing to search for possible implementation errors and search for the optimal configuration for the pneumonia problem.
- Results and conclusions.

Palabras clave

- Análisis de textura
- Visión por computador
- Reconocimiento de patrones
- SVM
- Co-ocurrencia
- Python
- Análisis digital de imágenes
- Aprendizaje automático

Keywords

- Texture analysis
- Computer vision
- Pattern recognition
- SVM
- Co-ocurrence
- Python
- Digital image processing
- Machine Learning

Índice

Índice de figuras

Índice de Tablas

Capítulo 1

Introducción

Resumen: en este capítulo se expone la motivación que nos ha llevado a desarrollar este proyecto, así como los objetivos iniciales del mismo.

1.1. Motivación

Desde los inicios de la informática siempre se ha tratado de crear máquinas, cada vez más complejas, que fueran capaces de emular algunas de las características, funciones y maneras de procesar la realidad propias del ser humano. En este sentido, las computadoras vendrían a ser un contenedor del conocimiento del hombre salvo por el hecho de que disponen de una capacidad de cómputo infinitamente superior. Serían un auxiliar que posibilitaría el descubrimiento de nuevos avances científicos y artilugios técnicos. En palabras de Vannevar Bush:

"*Ciertamente, se deberá asegurar la delegación en las máquinas del arduo trabajo que requiere la compleja y detallada manipulación matemática de los datos, si deseamos que el cerebro de esos científicos quede libre para canalizarlo a tareas mucho más importantes que la mera transformación repetitiva de los datos según reglas preestablecidas*"[3].

Uno de los campos de la informática donde más palpable se hace lo dicho y que más se ha expandido en los últimos tiempos es la visión por ordenador, la cual definiríamos como "*una disciplina científica que incluye métodos para adquirir, procesar, analizar y comprender las imágenes del mundo real con el fin de producir información numérica o simbólica para que puedan ser tratados por un ordenador*"[1]. Tal y como los humanos usamos nuestros ojos

[1] https://es.wikipedia.org/wiki/Visión_artificial

y cerebros para comprender el mundo que nos rodea.

Y es dentro del área de la visión por computador donde encontramos el campo del reconocimiento de patrones gráficos, que a su vez, engloba el análisis de texturas como una de las ramas más prominentes, ya que es la responsable de multitud de hallazgos y aplicaciones tan importantes como: Diagnóstico de cáncer de piel y lesiones pigmentadas benignas (Green et al., 1991), recuperación de imágenes por color y textura (Veltkamp y Hagedoorn, 1999), diagnóstico de miocarditis (Ferdeghini et al., 1991), detección de características de los hielos polares (Sephton et al., 1994), clasificación de las rocas volcánicas por textura (Hernández, 1995), clasificación de formaciones vegetales (Gil et al., 1997a), identificación de personas mediante la textura del Iris y detección de residuos en determinados medios utilizando cámaras ópticas.

Claro está que implementar todas estas herramientas no es una tarea sencilla. Para lograrlo, hay que apoyarse en técnicas de clasificación basadas en aprendizaje automático, como redes neuronales o, en nuestro caso, SVM; y en técnicas de ADI (Análisis Digital de Imágenes), las cuales están relacionadas con la extracción de mediciones útiles, datos o información de un campo de la imagen utilizando dispositivos y sistemas automáticos.

1.2. Objetivos

El objetivo principal de este TFG es implementar un analizador de texturas que sea capaz de identificar la presencia de un determinado material en una imagen tan solo ateniéndonos a su textura. No tendremos en cuenta otras características como el color y la forma del material.

Para ello habrá, en primer lugar, que analizar digitalmente las imágenes de nuestro dataset y obtener muestras que nos sirvan posteriormente para entrenar nuestra máquina y clasificar las imágenes. Esta tarea se completará haciendo uso de métodos estadísticos, basados en el histograma de la matriz de grises de la imagen; de la matriz de co-ocurrencia de segundo orden, derivada del histograma de grises; y de 14 descriptores de fácil cálculo.

En segundo lugar, habremos de instanciar una SVM (Support vector machine), para entrenar y clasificar, que reciba como entrada los vectores de características de las muestras recogidas en la primera parte, o las propias componentes de la matriz de co-ocurrencia .

Por último, y como complemento u objetivo secundario, aplicaremos to-

do el sistema sobre un caso concreto de la realidad y realizaremos un estudio sobre la calidad de los resultados obtenidos.

1.3. Visión general del documento

El documento estará dividido en los siguientes capítulos que se muestran a continuación:

- Capítulo 2: se explicará todo el estudio previo a la implementación del sistema.
- Capítulo 3: en este apartado explicaremos todo lo relacionado con la gestión del proyecto y la administración del trabajo.
- Capítulo 4: describiremos el sistema en su totalidad a partir de un análisis detallado de cada una de sus partes.
- Capítulo 5: comportamiento del sistema sobre un problema concreto.
- Capítulo 6: conclusiones del trabajo y proyectos futuros.
- Capítulo 7: se expondrán las contribuciones de cada miembro del equipo al resultado final.
- Apéndice A: ejemplo de ejecución de la aplicación.
- Apéndice B: manual de usuario.

1.4. Motivation

Since the beginning of computing, we have always tried to create increasingly complex machines that were able to emulate some of the characteristics, functions and ways to understand properly reality as human being would. In this sense, computers would become a container of the knowledge of man, except by the fact that they have an infinitely superior computing capacity. They would be an auxiliary that would enable the discovery of new scientific advances and technical devices. In the words of Vannevar Bush:

"*Relief must be secured from laborious detailed manipulation of higher mathematics as well, if the users of it are to free their brains for something more than repetitive detailed transformations in accordance with established rules.*"[3].

One of the fields of computer science where what has been said is most palpable and has been expanded the most in recent times is computer vision, which we would define as "*an interdisciplinary scientific field that deals with how computers can be made to gain high-level understanding from digital images or videos. From the perspective of engineering, it seeks to automate tasks that the human visual system can do.*"[2].

And it is within the area of computer vision where we find the field of recognition of graphic patterns, which, in turn, encompasses the analysis of textures as one of the most prominent branches, since it is responsible for a multitude of findings and applications as important as: skin cancer and benign pigmented lesions diagnosis(Green and anothers,1991), recovery of images by color and texture (Veltkamp and Hagedoorn,1999), myocarditis diagnosis(Ferdeghini and others, 1991), detection of polar ice characteristics(Sephton and others, 1994), classification of volcanic rocks by texture(Hernández, 1995), classification of plant formations(Gil and others, 1997), identification of people through the texture of Iris and detection of waste in certain area using optical cameras.

Of course, implementing all these tools is not a simple task. To achieve this, we must rely on classification techniques based on machine learning, such as neural networks or, in our case, SVM; and in the techniques of digital image análisis which are related to the extraction of useful measurements, data or information from a field of the image using devices and automatic systems.

[2]https://en.wikipedia.org/wiki/Computer_vision

1.5. Objectives

The main objective of this paper is to implement a texture analyzer able to identify the presence of a certain material in an image only by adhering to its texture. We will despise other characteristics such as the color and shape of the material.

To do this, we will first have to digitally analyze the images of our dataset and obtain samples that will serve us later to train our machine and classify the images. This task will be completed using statistical methods, based on the histogram of the gray matrix of the image; by the second-order co-occurrence matrix, derived from the gray histogram; and 14 easily calculated descriptors.

In second term, we will have to instantiate an SVM (Support vector machine), to train and classify, that receives as input the characteristics vectors of the samples collected in the first part, or the own components of the co-ocurrence matrix.

Finally, as a complement or secondary objective, we would like to apply the whole system on a concrete case of reality and realize a small study on the quality of the results obtained.

1.6. General vision of the document

The document will be divided into the following chapters:

- Chapter 2: the entire study previous to the implementation of the system.
- Chapter 3: in this section we will explain everything related to the project management and administration.
- Chapter 4: we will describe the system at its entirety from a detailed analysis of each of its parts.
- Chapter 5: behavior of the system on a specific problem.
- Chapter 6: conclusions of work and future projects.
- Chapter 7: the contributions of each team member to the final result.
- Appendix A: execution example.
- Appendix B: user manual.

Capítulo 2

Trabajo previo

Resumen: En este apartado se exponen los fundamentos teóricos del proyecto y las tecnologías para desarrollarlo.

2.1. Estado del arte

Si hablamos del análisis de texturas, hay una figura que, sin lugar a duda, sobresale por encima del resto de investigadores dedicados a esta materia: Robert Haralick. Él fue quien marcó el gran hito en el desarrollo de los métodos de análisis de texturas al formalizar en 1973 el llamado Modelo Estadístico de segundo orden con su matriz de co-ocurrencia y 14 descriptores de textura en 4 direcciones fundamentales, el cual nos ha permitido tener una descripción más certera de las principales características de una textura. [5]

Pero, para situar los orígenes del análisis de texturas hay que remontarse a 1955, momento en el que Kaizer comienza los primeros estudios de la textura en imágenes empleando funciones de autocorrelación [1] [6]. Algo más tarde, en 1967, se reanuda la investigación en este área, esta vez de manos de R.Bixby, quien incorpora un enfoque basado en mallas de Markov de primer y segundo orden. [10]

Hasta ese momento, a pesar de que estas primeras investigaciones lograron cierto éxito, el arte quedó estancado durante un tiempo, ya que no se llegaron a definir modelos sólidos para abordar el problema de la textura. Como mucho, lo único que se logró fue obtener algunas propiedades especí-

[1] https://es.wikipedia.org/wiki/Autocorrelación

ficas como la rugosidad, (tosquedad) [2] y detectar los bordes de la imagen. Por ejemplo, Rosenfeld y Troy (1970) [2] describieron un procedimiento para obtener una medida de la tosquedad de la textura en la imagen que estaba basado en las diferencias entre los valores de los tonos de gris de los elementos adyacentes de la imagen y utilizaron además la autocorrelación de los valores de los tonos de gris. [12]

La gran revolución llegó con Haralick en 1973. A partir de este momento se incorporan distintos modelos, la mayoría inspirados en Haralick y con un enfoque basado en estadísticas de segundo orden. Un ejemplo de ello es Galloway (1975) [4], que introdujo otra variante del Modelo estadístico de segundo orden para calcular 5 descriptores en las direcciones 0^o, 45^o, 90^o y 135^o.

En adelante y hasta entonces se desarrollaron otros modelos y enfoques, pero podemos convenir que, generalmente, el preponderante es el de las matrices de segundo orden, al menos en los métodos que utilizan un modelo estadístico [3]. Desde este momento comienza la estadística de segundo orden, mientras que aquellos modelos de tercer orden o superiores son descartados por suponer un coste computacional demasiado elevado.

A partir de los 80 la mayor cantidad de avances se enfocaron en tratar de obtener nuevos descriptores que permitieran hacer aplicaciones en imágenes y afinar los ya existentes. Caben ser destacados los siguientes trabajos: Medición de textura basada en la entropía del espectro de potencia normalizado en el dominio de la frecuencia espacial (Jernigan y DÁstous, 1984), Análisis de la textura a partir de la geometría fractal (Peleg, et al., 1984) y Nuevos rasgos de textura a partir de establecer relaciones espaciales entre los píxeles originales (Barros de Andrade y Niero Pereira, 1989). [12]

Más tarde durante los 90 continua esta tendencia y nos encontramos con que He y Wang (1991) introducen el espectro de textura, lo que da lugar a otro enfoque estadístico de segundo orden con 10 descriptores con expresiones matemáticas complejas a partir de la definición del histograma de unidades de textura. [12]

Más recientemente vemos que una parte de las publicaciones que versan sobre el análisis de texturas son de un carácter netamente teórico. Por ejemplo: estudios sobre el estado del arte y taxonomías sobre los distintos modelos de textura. Así observamos investigaciones como las de Tuceyran

[2] En mecánica, la rugosidad es el conjunto de irregularidades que posee una superficie: https://es.wikipedia.org/wiki/Rugosidad_(mecánica)

[3] Frente al modelo geométrico y al modelo espectral

y Jain (1998) que presentan una taxonomía de varios modelos de textura que incluye métodos estadísticos, métodos geométricos, métodos basados en modelos y métodos de procesamiento de señales. O también, las de Gonzáles y Woods (2004), la cuales expresan que para el estudio de las regiones se han empleado muchos descriptores que genéricamente se agrupan como Simples, Topológicos y de Textura, aunque en la práctica es común usar de manera combinada tanto los descriptores de frontera como los de región y refieren que los tres enfoques principales para describir la textura de una región son los modelos estadísticos, el estructural y el espectral. Otra gran parte de las publicaciones son aplicaciones, sobre todo fuera del mundo académico, a nuevas áreas de la ciencia o la industria: La textura presente en el Iris del ojo humano (Daugman, 1993; 1995 y 2004) y en la yema de los dedos (Jain et al., 2000; 2001 y Ross, 2003) comienza a utilizarse en Biometría con el objetivo de encontrar soluciones en tareas como la autenticación y la identificación de personas. [12]

Hasta nuestros días el campo ha seguido evolucionando ininterrumpidamente, hasta el punto de que, aparentemente, el único límite es la imaginación u originalidad de cada persona que quiera utilizar este tipo de tecnologías. A pesar de ello, aún hay algunas cuestiones que permanecen abiertas, de las cuales, muchas, no se espera que sean resueltas en el corto o medio plazo. Un ejemplo de esto sería el problema de la inexistencia de una definición formal del término textura, o la cuestión sobre cual sería la combinación de descriptores de textura óptimos para un caso concreto.

2.2. Modelo estadístico de la matriz de co-ocurrencia

Para explicar que es la matriz de coocurrencia y cuál es su utilidad tenemos que remitirnos obligatoriamente al artículo de Robert Haralick de 1973: **Textural features for image classification**. En este texto el autor explica claramente cuál es la necesidad de crear un modelo estadístico que sea realmente capaz de describir una textura mediante sus características y además nos presenta la matriz de coocurrencia como la base de dicho modelo.

La primera reflexión en torno a esta necesidad se encuentra al comienzo del artículo, aquí, Haralick nos dice que el paso más difícil para clasificar la información que nos puede dar una imagen a través de su textura es definir un conjunto de características significativas que la definan. Y para ello, no basta con apoyarse, tan solo, en la matriz de grises, también hay que tener en cuenta las relaciones espaciales que los píxeles de la matriz pueden tener entre sí. Esto es así pues sabemos que la textura es una característica muy abstracta que no admite ninguna definición que no sea ambigua y, por su-

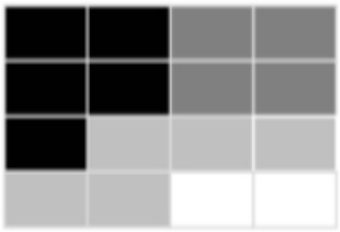

Figura 2.1: Ejemplo de escala de grises.

puesto, tampoco una definición matemática. En cambio, sí sabemos que es una característica íntimamente relacionada con el tono, el cual está basado en los tonos variables de las celdas de resolución de gris en la imagen fotográfica, mientras que la textura se refiere a la distribución parcial (estadística) de los tonos de gris. [5]

Es precisamente de esta relación de donde se extrae la idea de la matriz de co-ocurrencia de segundo orden. La llamamos de segundo orden porque, frente a las matrices de primer orden, es decir aquellas que únicamente tienen en cuenta los elementos de la matriz de forma individual sin considerar las posibles interdependencias que pueden existir entre los distintos elementos, la matriz de segundo orden sí extrae información de la asociación de elementos por parejas. Dicho de otra forma: considera la relación espacial entre dos píxeles, denominados píxel de referencia y píxel vecino.

Por tanto, definiremos a la matriz de coocurrencia como aquella que describe la frecuencia de un nivel de gris que aparece en una relación espacial específica con otro valor de gris, dentro del área de una ventana determinada. [9] Y se calculará como explicamos a continuación, con un ejemplo que tendrá como matriz de prueba la presentada en la Figura 2.1.

Para calcular una matriz de coocurrencia primero se nos darán unas direcciones, que pueden ser cualquiera de las 8 direcciones básicas (norte, sur, este, oeste y las 4 diagonales). Por ejemplo, escoger la dirección este, implica que el píxel vecino del píxel de referencia sea justo aquel que se encuentra a su derecha, por ello esta dirección se representará como (1,0).

También se nos dará una distancia, que en este caso viene a decir cuántas posiciones de diferencia hay en la dirección seleccionada entre el píxel de referencia y el vecino. Siguiendo el ejemplo anterior, si tenemos dirección este y distancia dos, representaremos la relación espacial como (2,0).

Pixel Vecino / Pixel de Referencia	0	1	2	3
0	(0,0)	(1,0)	(2,0)	(3,0)
1	(1,0)	(1,1)	(2,1)	(3,1)
2	(2,0)	(1,2)	(2,2)	(3,2)
3	(3,0)	(1,3)	(2,3)	(3,3)

Figura 2.2: Ejemplo de como resultaría la matriz de co-ocurrencia de distancia 1 y dirección este.

2	2	1	0
0	2	0	0
0	0	3	1
0	0	0	1

Figura 2.3: Matriz de co-ocurrencia (1,0) para la imagen de prueba.

Teniendo esto claro, el siguiente paso será componer una nueva matriz con todas las combinaciones de los cuatro niveles de gris de la imagen de prueba como se muestra en la Figura 2.2.

La primera celda debe ser llenada con la cantidad de veces que ocurre la combinación 0,0. Es decir, cuántas veces, en el área de la ventana un píxel con valor de gris igual a 0 (píxel vecino), está situado a la derecha de otro píxel con valor 0 (píxel de referencia).

Existen, por lo tanto, diferentes matrices de co-ocurrencia para cada relación espacial, según se considere el vecino de arriba, al costado o en diagonal. El resultado para la (1,0) se muestra en la Figura 2.3.

Sin embargo, para la co-occurrencia es necesaria la simetría y habrá que contabilizar por cada par de vértices, tanto que el uno se encuentre a la derecha del otro, como que el otro se encuentre a la izquierda del uno (Ob-

$$Pi,j = \frac{Vi,j}{\sum_{i,j=0}^{N-1} Vi,j}$$

Donde:
i es el numero de filas y j el numero de columnas
V es el valor de la celda i,j en la ventana
Pi,j es la probabilidad en la celda i,j
N es el numero de filas o columnas.

Figura 2.4: Normaliación de la matriz.

tenemos así el cálculo de ambos sentidos en una misma dirección, ahora tenemos: este-oeste, norte-sur, noreste-suroeste, noroeste-sureste). Esto se consigue sumando la matriz de coocurrencia calculada en una dirección a su traspuesta.

Ya que tenemos la matriz simétrica, lo siguiente y lo último que hay que hacer es expresarla como matriz de probabilidad, normalizarla tal como muestra la Figura 2.4.

Una vez que tenemos la matriz normalizada, podremos utilizarla para obtener medidas estadísticas, que describan la textura. Recordemos que, una vez que se definan estas características, los bloques de imágenes se pueden clasificar utilizando una de las múltiples técnicas de reconocimiento de patrones. Haralick definió estas 14 características: Segundo momento angular, contraste, correlación, varianza, momento de diferencia inversa, suma de promedios, suma de varianzas, suma de entropía, entropía, diferencia de varianza, diferencia de entropía, medidas de información de la correlación y máximo coeficiente de correlación. Cada característica nos dará información distinta, y será más relevante dependiendo de la naturaleza de la textura.

Las características que nosotros hemos implementado para formar los vectores de clasificación son las siguientes:

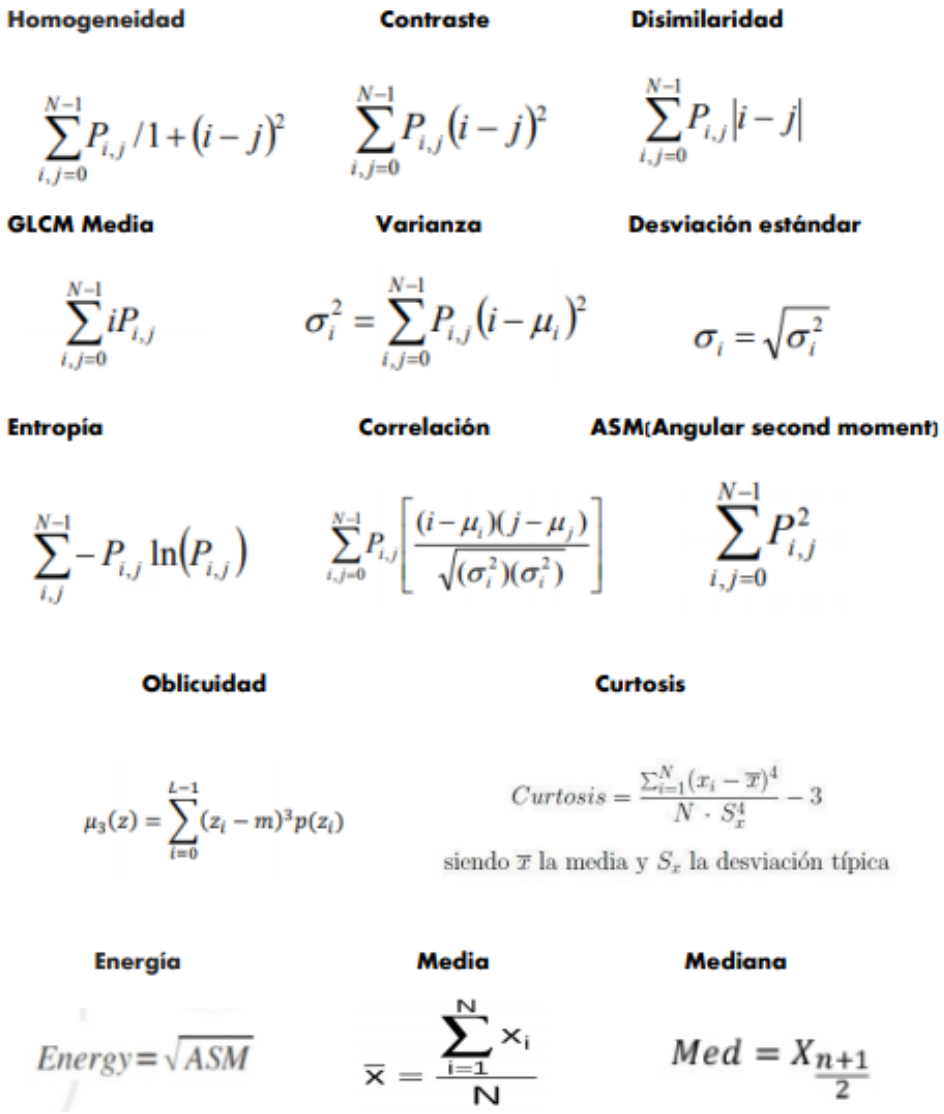

Figura 2.5: Características implementadas en el proyecto. Siendo P i,j la probabilidad de co-ocurrencia de los valores de gris i y j, para una distancia dada

2.3. SVM

Es bien sabido que existen una gran cantidad de algoritmos de clasificación, como las redes neuronales o la regresión logística, para clasificar conjuntos de datos. La mayoría de ellos funcionan de una forma muy simi-

lar, por lo que hay que tener un conocimiento muy elevado de los matices que aporta cada algoritmo para elegir el correcto en cada momento. La mayoría de las veces la variable más crítica de un problema será el tamaño de su conjunto de datos; la máquina de vector de soporte es una de las técnicas que más potencial puede aportar en esta situación.

Una descripción bastante aproximada de cómo funciona una SVM sería: *dado un conjunto de puntos, subconjunto de un conjunto mayor (espacio), en el que cada uno de ellos pertenece a una de dos posibles categorías, un algoritmo basado en SVM construye un modelo capaz de predecir si un punto nuevo (cuya categoría desconocemos) pertenece a una categoría o a la otra.* [4] Para completar esta definición, es necesario decir que la idea detrás de la máquina es la de clasificar un conjunto de datos de cualquier dimensionalidad, parametrizado matemáticamente, buscando el hiperplano que divide al conjunto de datos en las clases dadas. Siempre habrá muchos hiperplanos que clasifiquen a ese conjunto de datos de forma más o menos correcta. Por tanto, la clave es encontrar aquel que haga una mejor clasificación, es decir, aquél que deje un mayor margen entre ambas clases. Esta última es la razón por la que, a veces, se conoce a las SVM como clasificadores de margen máximo.

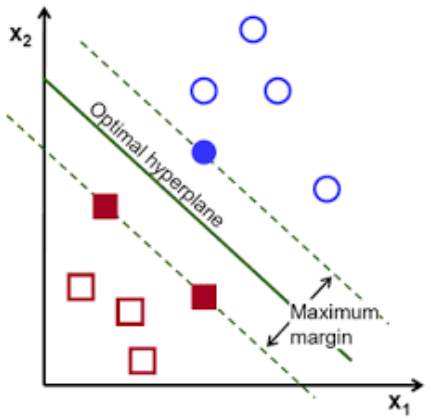

Figura 2.6: Concepto de la SVM.

Advertimos que en este trabajo no se profundizará en el aspecto matemático, pero puede encontrarse una descripción detallada en el libro *Support Vector Machines Succinctly by Alexandre Kowalczyk* [7].

[4] https://es.wikipedia.org/wiki/Máquinas_de_vectores_de_soporte

2.3.1. Maximal Margin Classifier e Hiperplano

En un espacio p-dimensional, un hiperplano se define como un subespacio plano y afín de dimensiones $p-1$. El término afín significa que el subespacio no tiene por qué pasar por el origen. [11] En el caso de un espacio de dimensión uno el hiperplano será un punto (divide una línea en dos). Si lo que tenemos es un espacio de dimensión dos, un plano, entonces el hiperplano será una recta. Esto es generalizable a cualquier dimensión mediante la siguiente ecuación:

$$\beta_0 + \beta_1 x_1 + \beta_2 x_2 + ... + \beta_p x_p = 0$$

Y es precisamente esta ecuación la que nos interesa para conocer el funcionamiento del SVM. Simplemente:

$$\beta_0 + \beta_1 x_1 + \beta_2 x_2 + ... + \beta_p x_p < 0$$

o bien

$$\beta_0 + \beta_1 x_1 + \beta_2 x_2 + ... + \beta_p x_p > 0$$

Vemos como el hiperplano divide el espacio en dos mitades, de tal forma que dependiendo del signo del resultado de resolver la ecuación para un punto X implica que este punto sea clasificado en una u otra clase. Lo dicho se comprenderá mucho mejor mediante un ejemplo.

En la siguiente imagen contemplamos un espacio bidimensional que es dividido mediante un hiperplano, en este caso una recta. La ecuación que describe esta recta es $1 + 2x_1 + 3x_2 = 0$. Así vemos que al área azul la conformarán los puntos que cumplen: $1 + 2x_1 + 3x_2 > 0$, mientras que la región roja será aquella que se forme con los puntos que satisfacen la misma ecuación con signo contrario.

Figura 2.7: Hiperplano de separación.

Para resolver los problemas de clasificación con SVM hay que encontrar el hiperplano óptimo de separación o *maximal margin hyperplane*, es decir aquel que se encuentra más alejado de todas las observaciones de entrenamiento. Para obtenerlo, se tiene que calcular la distancia perpendicular de cada observación a un determinado hiperplano. La menor de estas distancias determina el margen máximo de separación que habrá que optimizar hasta llegar al *maximal margin hyperplane*.

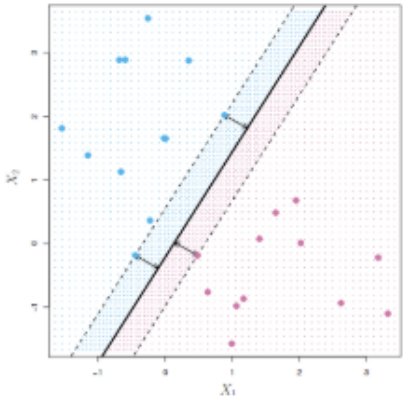

Figura 2.8: Hiperplano de separación máxima.

La imagen anterior muestra el *maximal margin hyperplane* para un conjunto de datos de entrenamiento. Las tres observaciones equidistantes respecto al *maximal margin hyperplane* se encuentran a lo largo de las líneas discontinuas que indican la anchura del margen. A estas observaciones se les conoce como vectores soporte, ya que son vectores en un espacio p-dimensional y soportan (definen) el *maximal margin hyperplane*. Cualquier modificación en estas observaciones (vectores soporte) conlleva cambios en el *maximal margin hyperplane* [11].

2.3.2. Clasificador de vector soporte o Soft Margin SVM

Siempre que clasificamos con SVM nos gustaría encontrar un hiperplano que fuera capaz de hacer una separación perfecta del conjunto de datos. Pero, a veces, esta separación o bien no es posible, o bien es contraproducente a la hora de generalizar el problema, al pasar de entrenamiento a clasificación.

En el primer caso se puede experimentar con distintos tipos de Kernels para adaptar "la forma"del hiperplano a "la nube de puntos". En el segundo caso estamos ante un problema de sobreajuste(overfitting), que solo se solu-

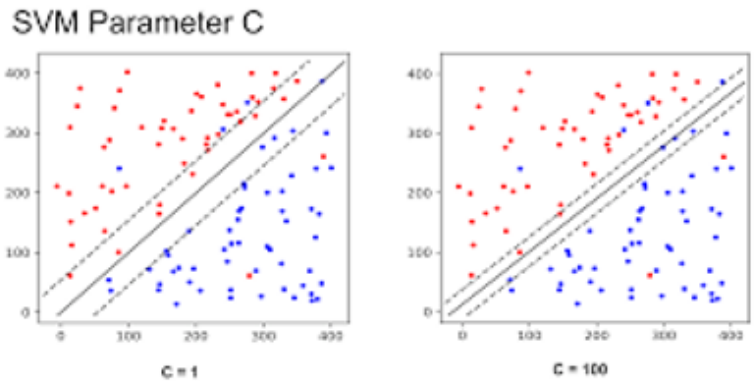

Figura 2.9: Clases no separable linealmente.

cionará usando un parámetro C que aporte cierta flexibilidad y *controle la compensación entre errores de entrenamiento y los márgenes rígidos, creando así un margen blando que permita algunos errores en la clasificación a la vez que los penaliza* [5]. Cuanto mayor es el valor de C mayor es el ajuste.

2.3.3. Kernels

En el punto anterior decíamos que, en ocasiones, la separación entre dos conjuntos de datos es inviable y que para ello necesitaríamos contar con otras formas de hiperplano que no clasifiquen de forma lineal. Esto se logra con la utilización de distintos tipos de kernel.

Un kernel es una función que devuelve el resultado del *dot product* entre dos vectores realizado en un nuevo espacio dimensional distinto al que se encuentran los vectores. Aunque no se ha entrado en detalle en las fórmulas matemáticas empleadas para resolver el problema de optimización, esta contiene un *dot product*. Si se sustituye este *dot product* por un kernel, se obtienen directamente los vectores soporte (y el hiperplano) en la dimensión correspondiente al kernel [11]. Existen multitud de kernels distintos, algunos de los más utilizados y que se emplean en el trabajo son:

- **Kernel lineal** (Figura 2.9): $K(x, x') = x * x'$. Si se emplea un Kernel lineal, el clasificador *Support Vector Machine* obtenido es equivalente al *Support Vector Classifier*.

[5] https://es.wikipedia.org/wiki/Máquinas_de_vectores_de_soporte

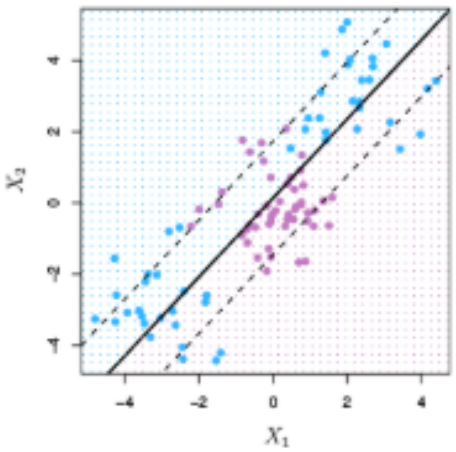

Figura 2.10: Kernel lineal.

- **Kernel polinómico** (Figura 2.10): $K(x, x') = (x * x' + c)^d$. Cuando se emplea d = 1 y c = 0, el resultado es el mismo que el de un *kernel* lineal. Si d >1, se generan límites de decisión no lineales, aumentando la no linealidad a medida que aumenta d. No suele ser recomendable usar d >5 por problemas de *overfitting*.

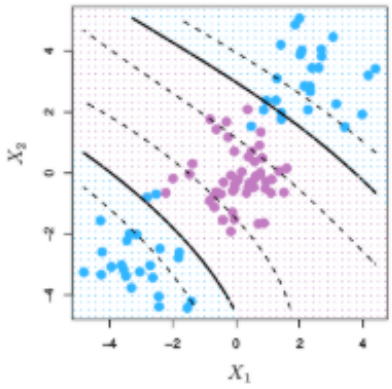

Figura 2.11: Kernel polinómico.

- **Kernel gaussiano** (Figura 2.11): $K(x, x') = exp(-\gamma||x - x'||^2)$. El valor de γ controla el comportamiento del kernel, cuando es muy pequeño el modelo final es equivalente al del kernel lineal, a medida que aumenta su valor, también lo hace la flexibilidad del modelo.

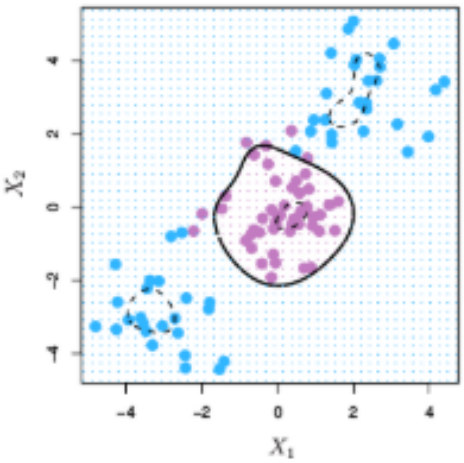

Figura 2.12: Kernel gaussiano.

2.4. Tecnologías relevantes

2.4.1. Python

Python es un lenguaje de alto nivel de los años 90 creado por Guido van Rossum que en los últimos años ha ganado bastante popularidad. Uno de sus grandes atractivos es que intenta acercarse al lenguaje natural, enfatizando la legibilidad y limpieza del código, permitiendo que sea relativamente sencillo realizar desarrollos usando dicho lenguaje.

También es un lenguaje interpretado que usa tipado dinámico y resolución dinámica de nombres. Soporta diferentes paradigmas, permitiendo que se trabaje en programación imperativa, funcional y orientada a objetos, lo que le da una gran versatilidad.

Python contiene una amplia librería estándar, y un gran número de módulos que le dan una gran cantidad de herramientas para trabajar en diferentes ámbitos, siendo uno de los más destacados el análisis de datos, lo cual hace que sea un lenguaje recurrente en el desarrollo de aplicaciones científicas.

Aunque finalmente no hemos usado las bibliotecas de Python para implementar las partes de aprendizaje automático de la máquina de vector de soporte, motivados por su facilidad de uso y por la existencia de herramientas para la elaboración de interfaces, sí lo hemos empleado en la construcción

de los módulos 1 y 3.

2.4.2. Pillow

Pillow es una librería OpenSource para Python que añade una gran cantidad de funciones de uso común para trabajar con imágenes, como pueden ser: aplicación de filtros, redimensionado de imágenes y cambios de formato.

Es una herramienta muy importante de cara al proyecto, dado que el primer y tercer módulo realizan un análisis donde se utilizan distintas operaciones para simplificar la extracción de datos sobre las múltiples imágenes.

2.4.3. NumPy y SciPy

NumPy es una librería OpenSource para Python que añade funcionalidades para trabajar con vectores y realizar calculos cientificos. Basada en la librería Numeric, se creó para tener un código de fácil mantenimiento, optimizado para dar mayor eficiencia, y flexible en su uso.

NumPy nos aporta diferentes estructuras, más eficientes que las estándar de Python, para encapsular datos como vectores o matrices(como es el caso de la matriz de Co-Ocurrencia).

SciPy es una librería OpenSource para Python que añade una gran colección de algoritmos matemáticos y proporciona a los usuarios una interfaz capaz de procesar datos con la misma eficacia que herramientas como Matlab u Octave.

2.4.4. PyQt

PyQt es un binding de la biblioteca gráfica Qt para el lenguaje de programación Python. Fue desarrollado por Riverbank Computing.

Qt es un framework que permite desarrollar aplicaciones multiplataforma e interfaces gráficas. Desarrollado en 1992, nace parcialmente en un desarrollo de código abierto, pero no completamente libre. Fue utilizado en el desarrollo de KDE, actualmente uno de los escritorios más populares de GNU/Linux.

2.4.5. JavaFX

JavaFX es una plataforma software con la capacidad de producir aplicaciones web provistas de las características de una aplicación de escritorio.

Es la alternativa a Swing más usada debido a la flexibilidad de estilos en las componentes, lo cual se consigue mediante el empleo de hojas css. Por esta razón, hemos optado por JavaFX para producir una interfaz dinámica para el módulo 2 que sea capaz de adaptar sus estilos en las diferentes plataformas, y que además sea lo más simple y ampliable posible.

Cabe destacar que, al emplear lenguaje Java, ha sido bastante sencillo comunicarse entre procesos de la terminal y realizar llamadas a programas de libSVM, consiguiendo que un proceso hijo sea esperado por su padre en sólo unas pocas líneas de código.

2.4.6. libSVM

libSVM es una de las librerías en código abierto más conocidas para trabajar con máquinas de vector soporte. Su licencia es BSD, por lo que hemos podido usar su código C para compilar nuestras propias versiones de los programas de entrenamiento y predicción.

Estas librerías contienen un amplio repertorio de APIs en diferentes lenguajes, por ejemplo en Java, C, C++, Python, R, etc. También trae soporte para apps de android.

Entre sus funcionalidades destacan los clasificadores de máquina soporte (SVC), regresiones (SVR) y estimación de una distribución (SVM de una clase). Para nuestro trabajo nos enfocamos exclusivamente en los clasificadores de máquina soporte, en especial C-SVC, que trabaja con el hiper-parámetro C, explicado anteriormente en la teoría.

Otra característica muy importante, es que contiene un script llamado grid.py, que se emplea para trabajar con validación cruzada. La validación cruzada es una técnica capaz de conseguir resultados óptimos de los valores C y gamma para un determinado caso de prueba con sus predictores.

En cuanto a kernels, libSVM soporta el kernel lineal, polinomial, radial (gaussiano) y el kernel sigmoide. En la parte práctica veremos que los kernels más útiles para nuestras clasificaciones serán el kernel polinomial y el radial.

Capítulo 3

Metodología de trabajo

Resumen: En este capítulo describiremos todo lo relativo a la gestión del proyecto, metodologías de trabajo y fases de desarrollo. También explicaremos las herramientas de apoyo utilizadas para la consecución de estos propósitos.

Desde el comienzo de este proyecto hasta la finalización del mismo, con la redacción de esta memoria, se pueden advertir dos etapas claramente diferenciadas: investigación y desarrollo. En cada una de ellas las metodologías tratadas y las herramientas puestas en práctica son completamente distintas:

- **Fase de investigación**: La primera parte de la fase de investigación, tras la primera reunión, consistió en un trabajo similar para todos los miembros del grupo. Cada uno de nosotros estudió en profundidad la bibliografía aportada por el profesor, la cual consiste mayoritariamente en trabajos del estado del arte del análisis de texturas, publicaciones universitarias, que explican los conceptos más importantes de la materia, y otros trabajos donde se experimenta con distintas técnicas para la clasificación de texturas. Además, complementamos el estudio de este material con información obtenida de Internet (principalmente Wikipedia y espacios dedicados al campo del machine learning con repositorios de imágenes como Kaggle [1], los repositorios de la universidad de california en Irvine [2] y Coursera [3]).

  Al final de este periodo tuvimos una segunda reunión en la que discutimos sobre las herramientas que utilizaríamos para llevar a cabo este

[1] www.kaggle.com
[2] http://archive.ics.uci.edu/ml/index.php
[3] https://www.coursera.org/learn/machine-learning

proyecto y sobre cómo abordarlo.

La segunda mitad de la investigación supuso una especialización de cada miembro en un aspecto del proyecto. Optamos porque una persona se dedicara al estudio del SVM y la otra investigara sobre técnicas ADI(análisis digital de imágenes) y procedimientos para obtener información sobre la textura de la imagen.

Cabe añadir que en la fase de investigación no hemos utilizado ningún tipo de software de control o gestión.

- **Fase de desarrollo**: En la fase de desarrollo pusimos en práctica lo aprendido en la investigación para desarrollar el sistema, el cual expondremos con detalle en el capítulo 4. Aunque cada uno de nosotros se centró en un módulo en concreto, el trabajo no estuvo exento de interdependencias que conseguimos superar gracias al software auxiliar de gestión.

3.1. Google drive

Google Drive es un servicio de almacenamiento de información en la nube, operativo desde 2012. Esta herramienta, que permite un almacenamiento gratuito de 15 GB para cada uno de sus usuarios, es de gran utilidad para visualizar y editar documentos entre varias personas a la vez.

Por ello, comenzamos a utilizar este software cuando fuimos conscientes de la cantidad de información que tendríamos que manejar para redactar la memoria y cuando quisimos empezar a dejar reflejado por escrito todos los pensamientos e ideas que estimábamos relevantes de plasmar en la memoria final.

3.2. Google hangouts

Las comunicaciones durante todo el proyecto se han basado en el uso de mensajería móvil y correos electrónicos. Pero en el momento de mantener reuniones a distancia para tratar aspectos técnicos de la aplicación, hemos optado por Google hangouts; una aplicación de mensajería, lanzada el 15 de mayo de 2013, la cual permite videollamadas y chats de grupo.

3.3. Gitlab

Es un servicio web de control de versiones y desarrollo de software colaborativo basado en git. Al igual que otras plataformas similares, como GitHub o Bitbucket, Gitlab ofrece una interfaz web con el fin de facilitar a los usuarios la gestión de repositorios a través de una experiencia más visual. Esta interfaz también funciona como una red social en la que desarrolladores de cualquier parte del mundo pueden intercambiar conocimientos y colaborar en proyectos. Aunque Gitlab comenzó como un proyecto de software libre bajo la licencia MIT, actualmente, la versión para empresas, la cual posee características adicionales, opera con licencia propietaria.

Pese a ofrecer un servicio semejante a Github, la herramienta más usada comúnmente, Gitlab nos ofrece mantener nuestro repositorio, y el de todas las cuenta básicas, privado sin requerir de ningún extra por ello.

3.4. GitKraken

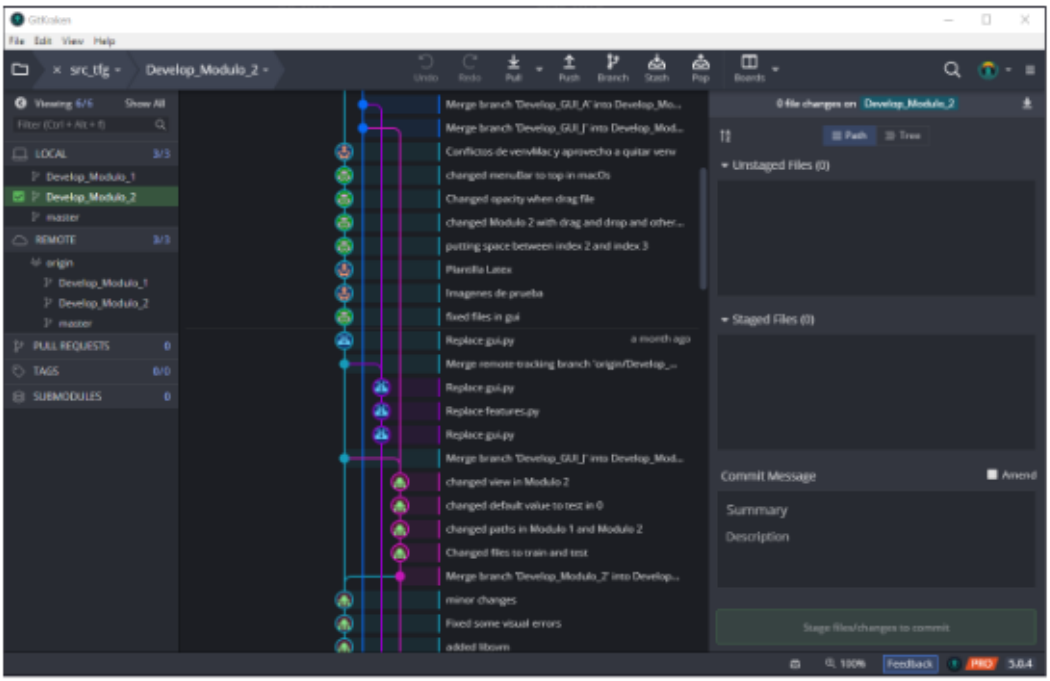

Figura 3.1: Vista de las ramas del proyecto desde GitKraken.

GitKraken es una potente interfaz gráfica para trabajar con git. Desarrollada en 2014 por Axosoft y puesta en el mercado en 2016, con el objetivo de ser una aplicación multiplataforma que popularice el uso de GUIs para git, esta herramienta ha resultado ser de gran utilidad a mucha gente para llevar el seguimiento del estado de los repositorios y para monitorizar el trabajo de los miembros de un proyecto, esto último gracias a la integración de Glo Boards.

Capítulo 4

Identificación de análisis por textura utilizando aprendizaje automático

Resumen: En este capítulo describiremos el sistema en su totalidad a partir de un análisis detallado de cada una de sus partes.

4.1. Funcionamiento general del sistema

Los módulos creados durante el proyecto están inspirados en los diferentes estudios alrededor de la capacidad de análisis de imágenes por texturas. En ellos se expone que características y que estructuras nos permiten obtener información relevante de las imágenes para, tras realizar un aprendizaje, poder realizar una identificación.

El proyecto se divide en tres módulos: el primero es un módulo de toma de muestras, el segundo módulo es un SVM con GUI para realizar el aprendizaje sobre los datos extraídos previamente del módulo uno y, finalmente, el tercero es un identificador que mediante los datos aprendidos identifica las imágenes que se le pasen para aplicar el aprendizaje realizado.

4.2. Módulo 1: Toma de muestras

El primer módulo sirve para obtener muestras, de forma binaria, a partir de un dataset de imágenes. Cada muestra será un recuadro de la imagen con una dimensión seleccionada por el usuario que se almacenará internamente como una matriz de píxeles. Es una de las partes más importantes a la hora de trabajar, pues el aprendizaje realizado en el segundo módulo vendrá

condicionado por el tratamiento que se le haya dado a las imágenes en este componente.

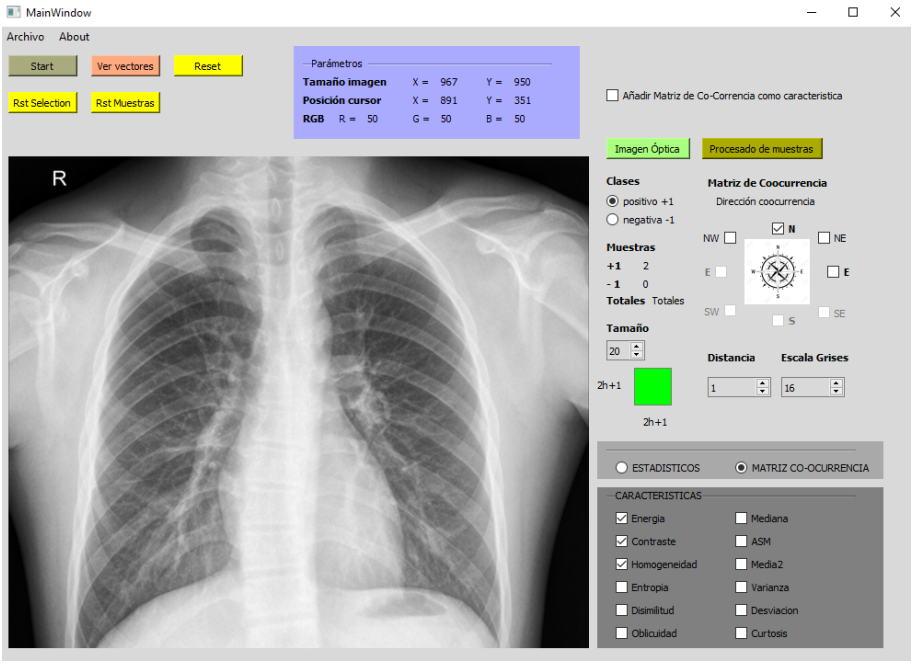

Figura 4.1: Vista del módulo de toma de muestras.

El módulo de toma de muestras cuenta con dos "modalidades"para realizar el análisis sobre la imagen: análisis estadístico y análisis por matriz de co-ocurrencia, construida tal y como indicamos en el capítulo 2 de esta memoria . En caso de querer realizar un análisis estadístico, el usuario seleccionará ESTADÍSTICOS, las características que desee aplicar a la imagen y procederá con el muestreo. Si lo que queremos es realizar un análisis sobre la matriz de co-ocurrencia, seleccionaremos MATRIZ CO-OCURRENCIA, las características deseadas, la dirección de la co-ocurrencia (es necesaria al menos una dirección), un escalado de grises(cada muestra será transformada a una matriz de grises), una distancia y, finalmente, se procederá al muestreo. De manera opcional se podrá marcar la opción .Añadir matriz de co-ocurrencia como característica", con ella podremos añadir cada componente de la matriz de co-ocurrencia como una componente más del vector de características de cada muestra analizada. En resumen podemos incorporar una estrategia más (comúnmente utilizada en estudios precedentes) para obtener información de las imágenes.

Ambas modalidades operarán sobre las características definidas en el capítulo 2, sin embargo, la primera de ellas analizará las muestras aplicando

filtros y cálculos directamente sobre los datos de las matrices de las muestras, mientras que la segunda calculará, en primer lugar, las matrices de co-ocurrencia de las distintas muestras, con la configuración seleccionada para dichas matrices, y aplicará las características sobre ellas, en vez de directamente sobre las muestras.

Finalmente queda el muestreo, que consistirá en seleccionar una de las dos clases de muestra (positiva o negativa), e ir clickeando sobre la imagen para que se marquen las muestras. Una vez terminado el muestreo se realiza el procesado de cada muestra y se almacena en una estructura de datos.

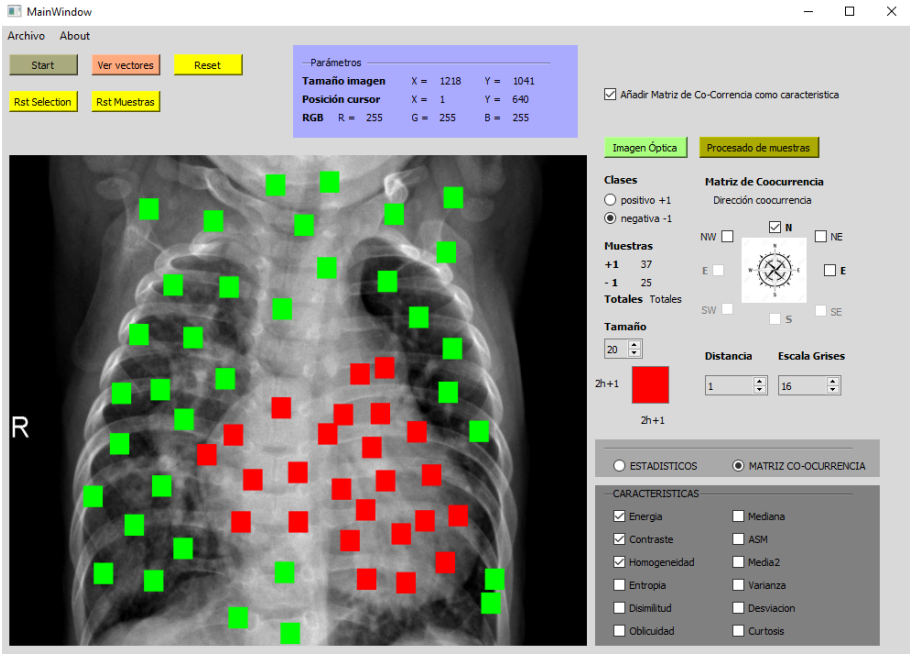

Figura 4.2: Ejemplo de muestreo.

Recordamos que la idea de esta aplicación es crear un un entorno operativo que facilite el desarrollo de diferentes aplicaciones de clasificación de texturas, permitiendo ensayos con diferentes parámetros. Como consecuencia de ello, y al ser un módulo de toma de muestras manual, con el objetivo de agilizar esta tarea, se ha tratado de implementar todo tipo de facilidades para la persona que interactúa con la aplicación: precauciones para que las muestras no se sobrepongan, restricciones para que las muestras no excedan los bordes, posibilidad de obtener muestras sobre distintas imágenes, etc.

4.2.1. Componentes del módulo

La elección de Python para el Clasificador viene dada por ser un lenguaje que destaca por su cantidad de paquetes que enfocan al análisis de datos. En este módulo se han hecho uso los siguiente paquetes de Python:

- PIL: este paquete de Python permite trabajar de forma sencilla con imágenes
- PyQt: este paquete es una implementación en Python de Qt un famoso framework GUI.
- SciPy: es un ecosistema de paquetes para realizar computación de datos científicos en Python.

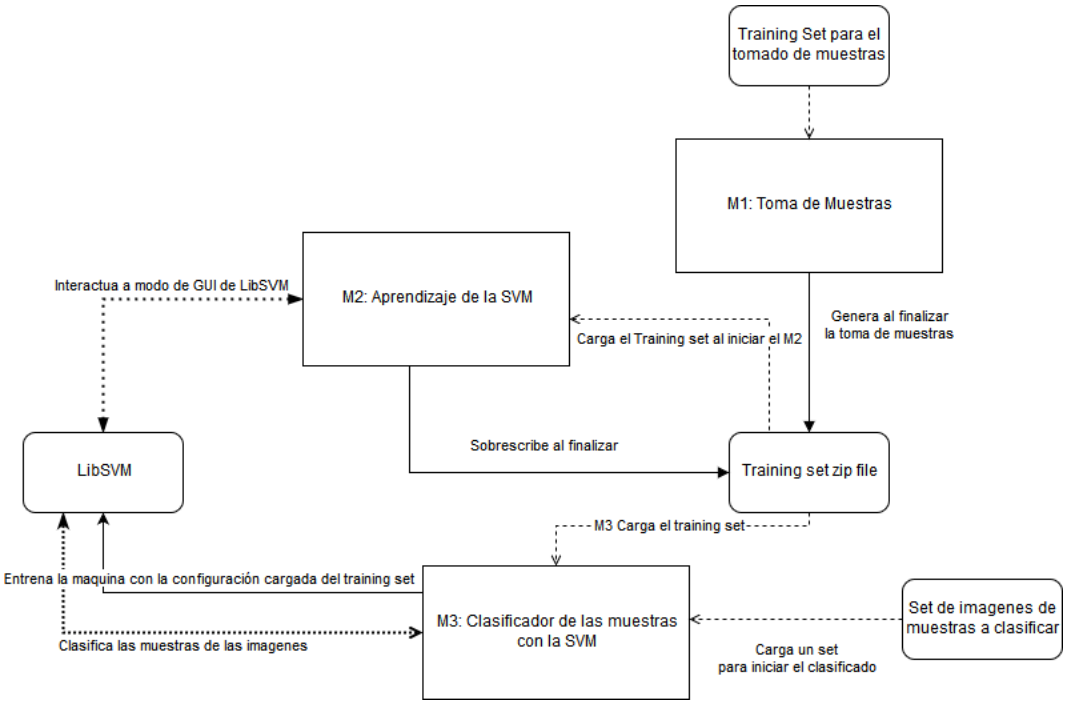

Figura 4.3: Diagrama de bloques del sistema

4.3. Módulo 2: Aprendizaje de la SVM

En este módulo, la aplicación se centra en la clasificación de los datos obtenidos por la salida del componente anterior, esto es, un archivo zip con un fichero para el entrenamiento con los vectores arrojados por el módulo 1 y un archivo de configuración json de cara a poder usar el módulo 3.

4.3.1. Componentes del módulo

Los archivos usados para este módulo son:

- svm-train: Es un ejecutable construido sobre C, cuya función es la de entrenar el dataset que le introducimos y generar un archivo model, que será aquel que use la máquina para luego poder realizar las clasificaciones.
- svm-predict: Otro ejecutable que clasifica el dataset de test y crea un porcentaje de aciertos entre las clases que estima la máquina y las clases que realmente representan los ejemplos. Además crea un fichero output que guarda todas las clases que estimó la máquina durante la clasificación.
- grid.py: Script de python opcional, pero útil para realizar K-fold cross-validation sobre el dataset. Para poder usarlo al máximo rendimiento, es imprescindible tener gnuplot instalado en el sistema, debido a que genera una imágen que representa una gráfica sobre los parámetros C y gamma con mejores resultados.

Para seleccionar qué tipo de clasificación se quiere realizar, en la interfaz gráfica se pueden seleccionar 5 tipos:

- C-SVC: Habilitado por defecto, debido a que es el más empleado en este trabajo. Equivale al explicado en la sección SVM Soft Margin en el capítulo 2.
- nu-SVC: Tiene la misma función que C-SVC, con la diferencia de que éste emplea el parámetro nu en lugar del parámetro C.
- SVM one class: Idóneo para poder estudiar la relación que tienen las características sobre una clase en específico.
- Epsilon SVR: Emplea regresión lógica sobre el dataset empleando el parámetro épsilon. Este clasificador nunca se emplea, debido a que el más idóneo para este trabajo es SVC.
- nu-SVR: Igual que el anterior, solo que empleando el parámetro nu en lugar de C.

Kernels utilizables (también explicados en el capítulo 2):

- lineal.

- polynomial.
- radial.
- sigmoid.

Los parámetros que se pueden emplear para realizar el entrenamiento de la máquina son:

- Degree: Define el parámetro d para el kernel polinomial.
- Gamma: Define el parámetro gamma para el kernel polinomial, radial y sigmoidal.
- Coef0: Define el parámetro r para el kernel polinomial y sigmoidal.
- Cost: Define el parámetro C para C-SVC y Epsilon SVR.
- Nu: Define el parámetro nu para nu-SVC, SVM one class y nu-SVR.
- Epsilon-SVR: Define épsilon para Epsilon SVR.
- Cache Size: Define el tamaño de la memoria caché para realizar la clasificación.
- Epsilon: Determina tolerancia para el criterio de terminación.
- Shrink: Activa la opción de acortar el número de iteraciones que realiza la máquina para aprender durante el entrenamiento.
- Estimate: Activa la opción de usar probabilidades estimadas de acierto generadas por un k-fold cross validation utilizado durante el entrenamiento. Es recomendable sólo usarlo una vez que se definen los parámetros óptimos para un dataset.
- Weight: Es un parámetro que actualiza C multiplicándose por el propio C.

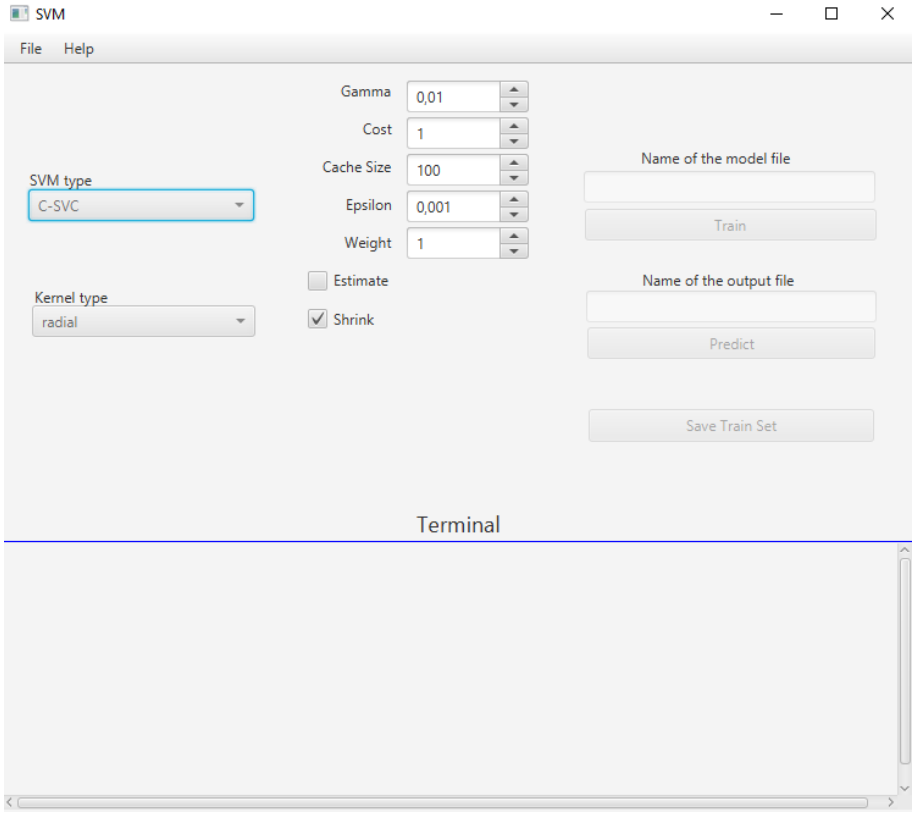

Figura 4.4: Vista del módulo 2

4.4. Módulo 3: Clasificación de muestras con la SVM

La idea de este módulo es realizar un barrido de una imagen dada con un tamaño de muestra y unas características idénticas a las utilizadas en el módulo de aprendizaje. Para ello se aplicarán las dos partes precedentes del sistema con el objetivo de obtener el vector de características de cada muestra del barrido. Finalmente se procederá a la clasificación de cada uno de estos vectores.

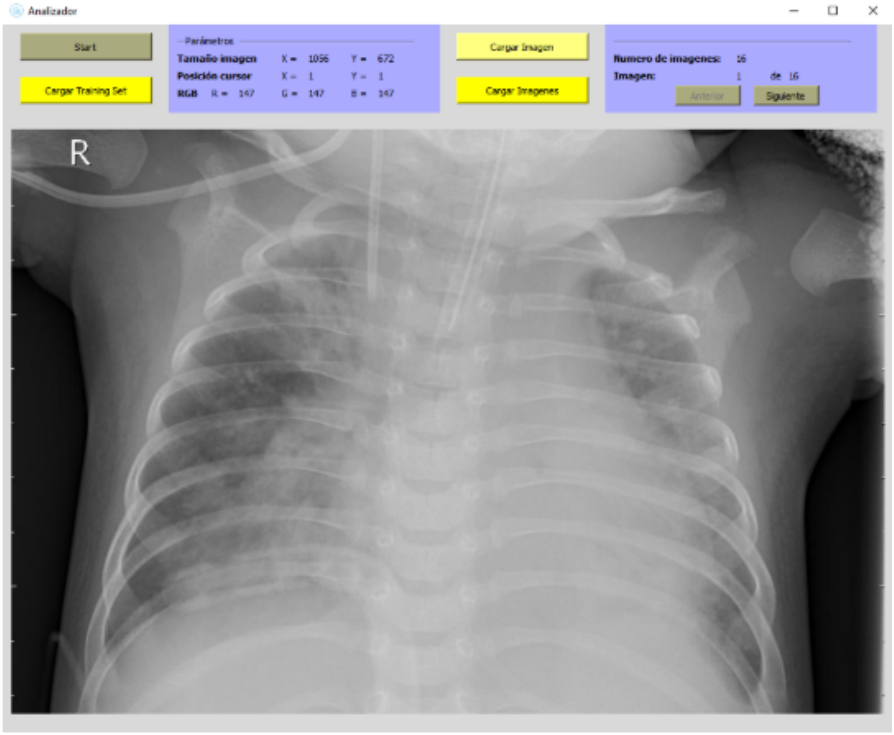

Figura 4.5: Vista del módulo 3

Para completar el barrido lo primero que habrá que hacer será cargar una imagen, o un conjunto de imágenes, junto con el training set arrojado por la SVM, es decir, un directorio comprimido, con extensión .zip, que contendrá un archivo de extensión .train con las muestras usadas para el entrenamiento de la SVM y un json con los parámetros de configuración que fueron seleccionados en el módulo anterior para entrenar a la máquina de vector soporte.

Una vez obtenidos los datos de entrada necesarios para que este componente funcione, se inicia un proceso que finalizará con las imágenes de entrada marcadas de tal forma que se diferenciarán, del resto de formas de la imagen, los objetos que queríamos adivinar por su textura. En el caso de la búsqueda de neumonía en radiografías de rayos x, las partes de la imagen que no son neumonías quedarán enmarcadas dentro de un recuadro rojo y las que sí lo son aparecerán sin dibujar.

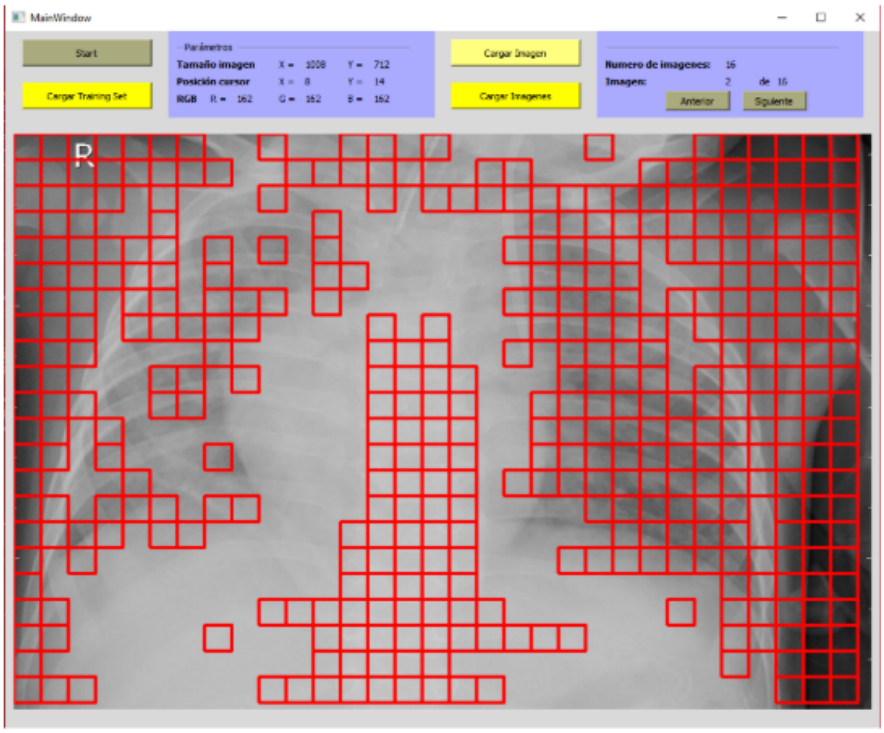

Figura 4.6: Ejemplo de resultado del barrido

El proceso se iniciará cargando la configuración del json y el archivo .train para inmediatamente después generar un comando que llame al módulo dos(a través de una tubería) y le pase tanto la configuración como las muestras del entrenamiento. Esto se hace con el fin de adquirir un modelo, tal y como el de la sección anterior, que sirva para clasificar las muestras del barrido.

Una vez que hemos entrenado a la máquina de nuevo y hemos obtenido el modelo de entrenamiento, procederemos a dividir la imagen en un grid, cuyas celdas tendrán el tamaño de muestra que se especificó en el módulo 1, y a recorrerlo de forma descendente, de izquierda a derecha. En este recorrido se clasificarán las muestras según dictamine el modelo y se generará una matriz de booleanos que se utilizará, finalmente, para pintar la imagen acorde al resultado de la clasificación.

Hay que aclarar que, en este módulo, para poder clasificar hay que crear un fichero .t con cada muestra y ejecutar, a partir de llamadas al sistema operativo(para acceder a libsvm), el método predict residente en el modelo de entrenamiento obtenido:

```
testCommand: str
if os == 'linux':
    testCommand = "../../libsvm/linux/svm-predict"
elif os == 'mac':
    testCommand = "../../libsvm/mac/svm-predict"
elif os == 'windows':
    testCommand = "..\..\libsvm\windows\svm-predict.exe"
```

Figura 4.7: Comando de acceso a libsvm

4.4.1. Componentes del clasificador

- El módulo subprocess: permite invocar procesos desde Python y comunicarse con ellos: enviar datos a la entrada (stdin) y recibir la información de salida (stdout). Además, esperar a que el proceso finalice o bien terminarlo prematuramente, y obtener el valor de retorno. [1]
- Al igual que en el módulo 1, también utilizamos las librerías PIL, numPy y PyQt.

[1] https://recursospython.com/guias-y-manuales/subprocess-creacion-y-comunicacion-con-procesos/

Capítulo 5

Pruebas y evaluación

Resumen: En este capítulo mostraremos el comportamiento del sistema sobre un problema concreto.

5.1. Introducción

Desde que comenzamos a trabajar en el sistema de identificación de imágenes por análisis de texturas nos hemos estado preguntando cual sería el campo de investigación más apropiado para aplicar nuestro desarrollo.

El primer estudio que consideramos hacer fue el de aplicar el análisis de textura al diagnóstico de enfermedades odontológicas, pero tras hablar con un odontólogo y obtener un conjunto de datos para el entrenamiento y la clasificación, decidimos descartar esta investigación, ya que no encontramos la forma de aplicar nuestro sistema al dataset recibido.

Tras este primer intento, la osteoporosis fue el siguiente problema que valoramos. Contactamos con la universidad politécnica de Orleans (Francia) para que nos legaran su dataset etiquetado con muestras de dicha enfermedad, pero las imágenes que nos dieron eran válidas, en el mejor de los casos, para resolver un problema de clasificación, no para la identificación de un objeto dentro de la imagen, tal y como pretendemos nosotros.

Finalmente hemos decidido emplear nuestra aplicación en la identificación de la neumonía a partir de las radiografías de la caja torácica de distintos pacientes. El conjunto de datos para este experimento se ha obtenido de Kaggle, que a su vez lo ha obtenido del trabajo "**ChestX-ray8: Hospital-scale Chest X-ray Database and Benchmarks on Weakly-Supervised Classification and Localization of Common Thorax Diseases**",[13] el cual realiza un estudio muy similar al nuestro, aunque utiliza

técnicas completamente distintas.

5.2. El problema

La neumonía es una enfermedad del sistema respiratorio que consiste en la inflamación de los espacios alveolares de los pulmones. [1]. Es una de las enfermedades que más muertes causan en todo el mundo a lo largo de año, siendo los niños y las personas mayores de 65 años los grupos de edad más afectados. Según el informe de 'Defunciones según la causa de muerte'del año 2015, difundido por el Instituto Nacional de Estadística(INE) [2] tan solo en nuestro país se registraron en ese año 10.209 víctimas por neumonía, un 20.9 por ciento más que en el año anterior, cifras que son mucho mayores en aquellos países con menos recursos sanitarios.

Su diagnóstico se realiza, la mayoría de las veces, a partir de una radiografía de tórax con el fin de ubicar la infección, comprobar su magnitud y, en algunos casos, tratar de discernir si su origen es bacteriano o vírico, lo cual es necesario si de lo que se trata es de salvar vidas, ya que en los lugares de menos recursos no siempre es posible una rápida interpretación de las radiografías.

Nuestro objetivo es mostrar la efectividad del sistema desarrollado detectando la neumonía en diversas radiografías de tórax.

5.3. El conjunto de datos

El conjunto de datos consta de un total de 5.232 imágenes de rayos X de niños, de ellas, 3.883 están etiquetadas como neumonía(2.538 neumonía bacteriana y 1.345 vírica) y 1.349 como sanas. El principal problema de este dataset es que las imágenes, aunque etiquetadas, no especifican con precisión el área del tórax en que se sitúa la neumonía. Por esta razón hemos complementado el dataset de kaggle con imágenes obtenidas de internet, que sí precisan cuál es la zona infectada, y además hemos consultado a un experto en la materia para que, sobre el conjunto de imágenes de kaggle, nos concrete los puntos exactos en los que aparece la neumonía.

Para poder realizar el experimento hemos reducido el dataset a 32 imágenes de las cuales serán 16 para el entrenamiento y 16 para el test, 16 tendrán

[1] https://es.wikipedia.org/wiki/Neumonía
[2] www.ine.es/prensa/edcm_2015.pdf

neumonía y 16 serán radiografías de un paciente sano.

5.4. El experimento

Con el propósito de acercarnos a una configuración óptima para el problema de identificación de la neumonía, hemos realizado una serie de experimentos y casos de prueba cuyos resultados valoraremos en función del porcentaje de acierto de la máquina con los test de entrenamiento y en la existencia de dos tipos de errores :

- **Falsos positivos**: Se trata de resultados que han sido devueltos incorrectamente, esto es, que se han reconocido como pertenecientes a la categoría consultada pero que no lo son.
- **Falsos negativos**: Son imágenes que no son reconocidas como pertenecientes a la categoría consultada cuando en realidad deberían ser aceptadas.

Casos de prueba	*Aciertos*	*Porcentaje de acierto*	*Falsos positivos*	*Falsos negativos*
Caso 1	*62/100*	*62%*	*38/38*	*0/38*
Caso 2	*86/100*	*86%*	*9/14*	*5/14*
Caso 3	*89/100*	*89%*	*5/11*	*6/11*
Caso 4	*83/100*	*83%*	*10/17*	*7/17*
Caso 5	*82/100*	*82%*	*10/18*	*8/18*
Caso 6	*80/100*	*80%*	*9/20*	*11/20*
Caso 7	*78/100*	*78%*	*16/22*	*6/22*
Caso 8	*172/200*	*86%*	*20/38*	*18/38*
Caso 9	*164/200*	*82%*	*26/46*	*20/46*
Caso 10	*59/100*	*59%*	*41/41*	*0/41*
Caso 11	*737/800*	*92%*	*33/63*	*30/63*
Caso 12	*678/800*	*85%*	*80/122*	*42/122*
Caso 13	*661/700*	*94%*	*17/39*	*22/39*
Caso 14	*1813/2000*	*90%*	*112/187*	*75/187*

Figura 5.1: Resultados de las pruebas

Configuraciones probadas en este experimento:

- **Caso 1:** tamaño de muestra: 50, distancia: 1, escala de grises: 16, kernel radial, matriz de estadísticos, todas las características seleccionadas, casos de entrenamiento: 200.
- **Caso 2:** tamaño de muestra: 50, distancia: 1, escala de grises: 16, kernel radial, matriz de estadísticos, energía y media2 seleccionadas, casos de entrenamiento: 200.
- **Caso 3:** tamaño de muestra: 50, distancia: 1, escala de grises: 16, kernel radial, matriz de estadísticos, energía seleccionada, casos de entrenamiento: 200.
- **Caso 4:** tamaño de muestra: 50, distancia: 1, escala de grises: 16, kernel radial, matriz de co-ocurrencia con todas las direcciones habilitadas, todas las características seleccionadas, casos de entrenamiento: 200.
- **Caso 5:** tamaño de muestra: 50, distancia: 1, escala de grises: 16, kernel radial, matriz de co-ocurrencia con la dirección este habilitada, -mediana, ASM, media2, varianza, desviación, curtosis y media- seleccionadas, casos de entrenamiento: 200.
- **Caso 6:** tamaño de muestra: 50, distancia: 1, escala de grises: 16, kernel radial, matriz de co-ocurrencia con la dirección este habilitada, -energía, contraste, homogeneidad, entropía, disimilitud, oblicuidad y correlación- seleccionadas, casos de entrenamiento: 200.
- **Caso 7:** tamaño de muestra: 50, distancia: 1, escala de grises: 16, kernel radial, matriz de co-ocurrencia con todas las direcciones habilitadas, -varianza, desviación y media- seleccionadas, casos de entrenamiento: 200.
- **Caso 8:** tamaño de muestra: 20, distancia: 1, escala de grises: 16, kernel radial, matriz de co-ocurrencia con todas las direcciones habilitadas, todas las características seleccionadas, casos de entrenamiento: 500.
- **Caso 9:** tamaño de muestra: 20, distancia: 1, escala de grises: 16, kernel radial, matriz de co-ocurrencia con la dirección este habilitada, -mediana, ASM, media2, varianza, desviación, curtosis y media- seleccionadas, casos de entrenamiento: 500.
- **Caso 10:** tamaño de muestra: 20, distancia: 1, escala de grises: 16, kernel radial, matriz de co-ocurrencia con todas las direcciones habilitadas, todas las características seleccionadas, casos de entrenamiento: 200.
- **Caso 11:** tamaño de muestra: 16, distancia: 8, escala de grises: 16, kernel radial, matriz de co-ocurrencia con todas las direcciones habilitadas, -homogeneidad, disimilitud, correlación, varianza y desviación-

seleccionadas, casos de entrenamiento: 1500, usamos la matriz de co-ocurrencia como componente del vector.

- **Caso 12:** tamaño de muestra: 32, distancia: 8, escala de grises: 16, kernel radial, matriz de co-ocurrencia con todas las direcciones habilitadas, -homogeneidad, disimilitud, correlación, varianza y desviación- seleccionadas, casos de entrenamiento: 1500, usamos la matriz de co-ocurrencia como componente del vector.

- **Caso 13:** tamaño de muestra: 16, distancia: 8, escala de grises: 8, kernel radial, matriz de co-ocurrencia con todas las direcciones habilitadas, -contraste, homogeneidad, entropia, disimilitud, correlación, ASM, varianza y desviación- seleccionadas, casos de entrenamiento: 1500, usamos la matriz de co-ocurrencia como componente del vector.

- **Caso 14:** tamaño de muestra: 20, distancia: 10, escala de grises: 10, kernel radial, matriz de co-ocurrencia con todas las direcciones habilitadas, -contraste, homogeneidad, entropia, disimilitud, correlación, ASM, media2, varianza y desviación- seleccionadas, casos de entrenamiento: 3000, usamos la matriz de co-ocurrencia como componente del vector.

Con la mejor configuración, el caso 13, la máquina es capaz de delimitar, con un margen de error, la zona de mayor opacidad, es decir, la que tiene presencia de neumonía, del resto de la imagen. Algunos de los resultados obtenidos con estos parámetros son:

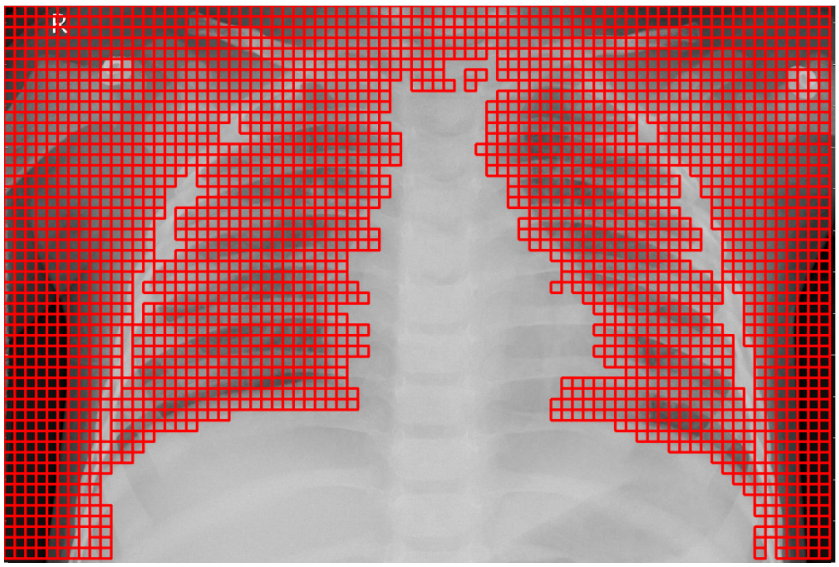

Figura 5.2: Barrido resultante de la configuración 13

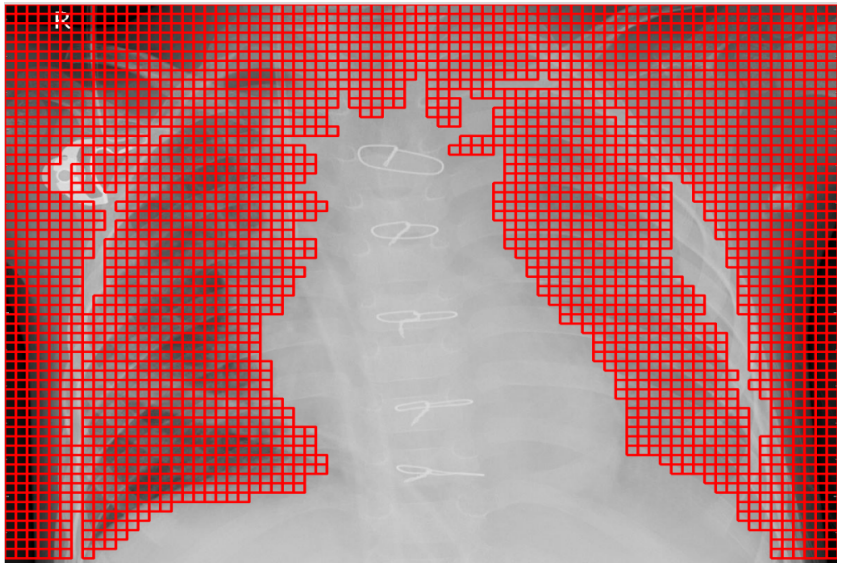

Figura 5.3: Barrido resultante de la configuración 13

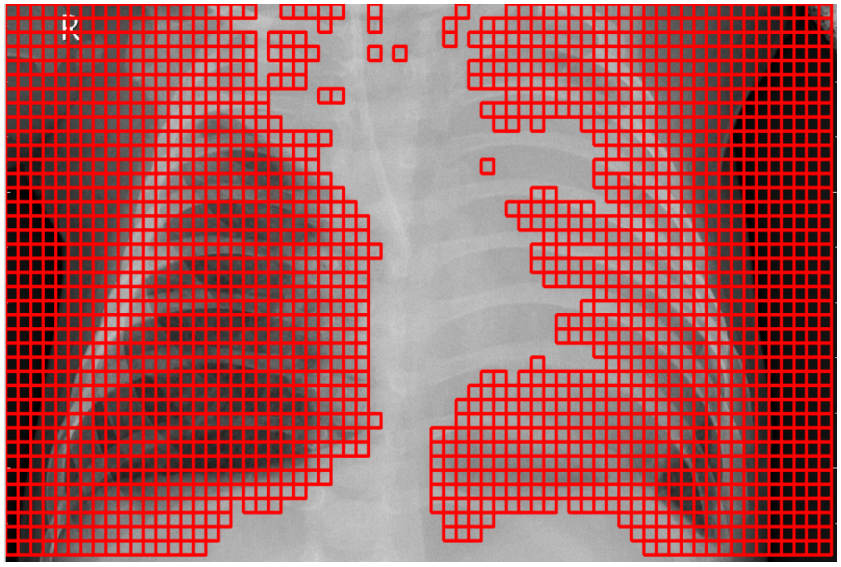

Figura 5.4: Barrido resultante de la configuración 13

Capítulo 6

Conclusiones y trabajo futuro

Resumen: En el presente capítulo expondremos algunas de las ideas, impresiones y conclusiones posteriores a la finalización el proyecto, así como estudios complementarios y posibles mejoras en cuanto a la eficiencia del sistema.

6.1. Conclusión

Creemos que para comenzar con este apartado de conclusiones lo mejor que podemos hacer es hablar honestamente acerca de qué es lo que más nos ha llamado la atención sobre este trabajo de fin de grado.

Diríamos que lo más impresionante para nosotros ha sido la sencillez de las técnicas aplicadas para resolver un problema tan complejo y necesariamente caótico como es identificar un objeto a través de su textura, ya que unas pocas funciones matemáticas elementales, un modelo de matriz, cuya formación no es compleja en absoluto, y alguna de las innumerables técnicas de aprendizaje automático para resolver problemas de regresión son requisitos suficientes para identificar una gran parte de las texturas, tanto naturales como artificiales.

Pero si las técnicas son tan sencillas, ¿Por qué cuándo indagamos en el estado del arte vemos que durante más de 50 años no han dejado de producirse avances y cambios en la materia? Nosotros creemos que esto se debe fundamentalmente a dos factores.

El primero de ellos es, obviamente, el aumento de la capacidad y la velocidad de procesamiento del hardware, dado que problemas de este tipo, que requieren de manejar una cantidad enorme de información con el fin de hacer

un retrato más fidedigno de la realidad, son muy susceptibles de incrementar la efectividad de los resultados paralelamente a la capacidad de cómputo.

El segundo de estos factores es la gran cantidad de variantes que hay para resolver un problema concreto. Y no solo eso, sino que cualquier pequeña modificación o combinación de distintos paradigmas puede, dependiendo del caso, arrojar resultados completamente distintos e incluso constituir una nueva variante. Por ello, los estudios del arte realizados por multitud de autores en el pasado son la base imprescindible para conseguir éxitos en este campo, ya que con ellos podemos tener un conocimiento general sobre sobre multitud de técnicas muy importantes.

En cuanto al desarrollo del proyecto, hemos de decir que estamos satisfechos con el resultado final debido a que, durante la realización de este, hemos adquirido unos valiosos conocimientos y una abundante experiencia gracias a que para implementar el sistema hemos tenido que entremezclar, de manera completamente libre y no reglada, distintos tipos de tecnologías y lenguajes de programación, lo que nos a forzado, hasta cierto punto, a ser creativos a la hora de inventar soluciones para los problemas que han ido surgiendo y a familiarizarnos con dichas herramientas.

También nos gustaría destacar que trabajar en este proyecto nos ha hecho concienciarnos de la utilidad de usar sistemas de control de versiones, de planificar correctamente un proyecto desde el principio y de la importancia de realizar una investigación previa completa.

En las siguientes páginas explicaremos lo que, a nuestro juicio, son posibles ampliaciones, complementos o variantes relevantes para nuestro proyecto actual.

6.2. Conclusion

We believe that to start this section of conclusions the best we can do is to speak honestly about what has most caught our attention about this final project.

We would say that the most impressive thing for us has been the simplicity of the techniques applied to solve a problem as complex and necessarily chaotic as it is to identify an object through its texture, being that a few elementary mathematical functions, a matrix model, whose formation is not complex at all, and some of the innumerable automatic learning techniques to solve regression problems are sufficient requirements to identify a large

part of the textures, both natural and artificial.

But if the techniques are so simple, why when we investigate the state of the art we see that for more than 50 years there have been progresses and changes in the matter? We believe that this is fundamentally due to two factors.

The first of these is, obviously, the increase in the capacity and speed of hardware processing, given that problems of this type, which require handling a huge amount of information in order to make a more reliable portrait of reality, they are very susceptible to increase the effectiveness of the results parallel to the computing capacity.

The second of these factors is the large number of variants that exist to solve a specific problem. And not only that, but any small modification or combination of different paradigms can, depending on the case, yield completely different results and even constitute a new variant. For this reason, the studies of art made by many authors in the past are the essential basis to achieve success in this field, since with them we can have a general knowledge about a multitude of very important techniques.

Regarding the development of the project, we have to say that we are satisfied with the final result. because during the realization of this, we have acquired valuable knowledge and an abundant experience thanks to the fact that to implement the system we have had to intermingle, completely free and not regulated, different types of technologies and programming languages, which We have been forced, to some extent, to be creative when it comes to inventing solutions for the problems that have arisen and to become familiar with these tools.

We would also like to highlight that working on this project has made us aware of the utility of using version control systems, of correctly planning a project from the beginning and of the importance of carrying out a complete prior investigation.

In the following pages we will explain what, in our opinion, possible extensions, complements or variants relevant to our current project.

6.3. Trabajo futuro

6.3.1. Utilizar otros métodos de análisis

En capítulos previos hemos visto que los métodos de análisis de texturas pueden ser clasificados de muchas formas, aunque generalmente se suelen dividir en: estadísticos, basados en la distribución espacial de los niveles de grises; estructurales, los cuales suponen que la textura está formada por una estructura que se repite, denominada primitiva; y espectrales.

Una de las vías que nos gustaría aplicar en el futuro sería combinar el actual método estadístico de la co-ocurrencia con métodos estructurales, concretamente utilizar la geometría fractal para encontrar patrones en la textura, en la línea del trabajo "*Análisis de la textura a partir de la geometría fractal*"(Peleg, et al., 1984)

Figura 6.1: Ejemplo de geometría fractal en la naturaleza.

La razón de esta idea surge de lo intuitivo que resulta aplicar el fractal para describir la disposición de las partículas de una textura. Mientras que la geometría euclídea clásica a menudo resulta insuficiente para describir las formas enrevesadas y caóticas que comúnmente apreciamos en los elementos más microscópicos, la geometría fractal como contraposición a esta resulta algo mucho más pertinente de aplicar en el mundo de las imperfectas texturas. [1]

[1]La matemática siempre ha servido de lenguaje de las ciencias, sobre todo de las llamadas por algunos fácticas. A partir de ellas se han construido modelos que simulan la realidad con mayor o menor éxito y que frecuentemente estos modelos sustituyen a la mis-

Además, utilizar fractales es una opción interesante puesto que son una de las formas más comunes de la naturaleza, *presentes en la materia biológica, junto con las simetrías (las formas básicas que solo necesitan la mitad de información genética) y las espirales (las formas de crecimiento y desarrollo de la forma básica hacia la ocupación de un mayor espacio).* [2]

6.3.2. Velocidad de ejecución

Una de las facetas que convendría mejorar de cara al futuro sería la velocidad con la que se ejecuta la aplicación. Pese a que ya hemos aplicado algunas técnicas para este propósito(multithreading), el margen de mejora sigue siendo bastante amplio.

Una técnica que cobra cada vez más importancia en las aplicaciones científicas y de cálculo intensivo es el uso de tarjetas aceleradoras de vídeo. Al fin y al cabo, este tipo de procesadores fueron diseñados para este tipo de trabajos y por tanto es una idea muy sensata utilizarlas en nuestro sistema.

Otra manera de mejorar la velocidad de ejecución sería plantear una optimización del código.

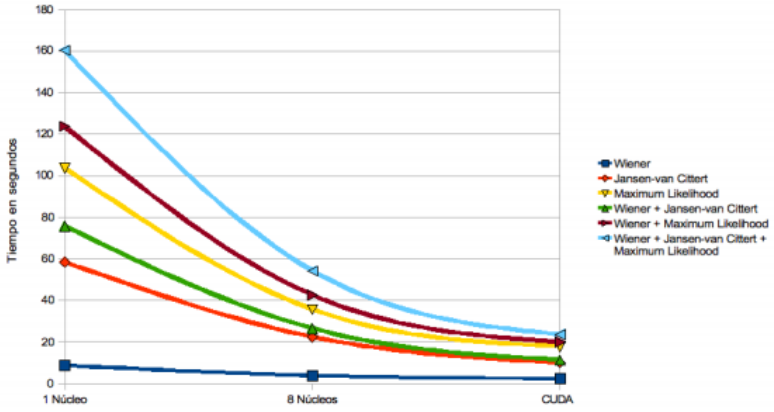

Figura 6.2: Comparativa de rendimiento en la deconvolución de una imagen, implementada sobre una CPU, multiprogramada sobre ocho núcleos y en una GPU mediante CUDA. [8]

ma realidad. En el caso de los procesos rugosos, roughness en el lenguaje de Mandelbrot, tales como pliegues, o bien fenómenos que se autoreplican como los proceso de reproducción de células, las matemáticas salvo modelos estadísticos y algunos analíticos han tenido un éxito escaso [1]

[2] https://es.wikipedia.org/wiki/FractalCaracterísticas_de_un_fractal

6.3.3. Otros trabajos futuros

- Experimentar con distintos tipos de hardware para la toma de muestras; cámaras térmicas, por ejemplo.
- Combinar el identificador de objetos con un clasificador. Por ejemplo, una vez automatizado el proceso de identificación de la neumonía, podemos complementar el diagnóstico implementando un clasificador que distinga entre neumonía vírica o bacteriana.
- Utilizar otro tipo de técnicas de aprendizaje automático distintas al SVM.

Capítulo 7

Contribuciones individuales

7.1. Nicolás Alcaine Camilli

Al comienzo del trabajo, se realizó una separación de módulos para que cada uno de nosotros pudiese encargarse prácticamente con alguno de ellos y especializarse, pudiendo realizar una investigación sobre él y posteriormente documentarlo en la memoria junto con la implementación del código necesario para llevarlo a la parte práctica.

De aquí en la primera reunión se eligió a Alejandro como encargado de todo el primer módulo, es decir, toda la parte de análisis de texturas en imágenes y su implementación. A mí se me asignó el segundo módulo, que se encarga de analizar todas las características estudiadas en el anterior módulo y poder realizar clasificaciones de objetos presentes en las imágenes usando aprendizaje automático, en concreto empleando SVM, debido a su gran capacidad de clasificación mediante sus kernels.

Al comienzo de la realización de mi módulo, tuve la idea de realizarlo bajo Python, debido a que entre nosotros habíamos visto que Python es un lenguaje que poco a poco se fue especializando en las matemáticas y en particular, sobre data science (dentro de éste también en aprendizaje automático). También mi idea fue esa porque tenía pensado unificar ese módulo junto con los demás y ellos ya los habían empezado a implementar bajo ese lenguaje.

Más tarde procedí a ir implementando el kernel gaussiano bajo la clasificación usando el parámetro C, pero no continué por una posterior reunión con el tutor, donde habíamos decidido entre nosotros 2 que el módulo iba a estar completamente separado del resto de módulos y que se podría usar directamente una librería capaz de realizar todas las operaciones bajo SVM.

Al final opté por usar JavaFX como lenguaje de mi módulo, debido a su facilidad y moldeabilidad a la hora de crear interfaces de usuario. Anteriormente sólo había trabajado con el lenguaje Java, pero se me hizo bastante fácil su aprendizaje ya que también realizé trabajos de páginas web con CSS3 en el pasado, y toda la parte gráfica de sus elementos se definen bajo propiedades CSS.

También opté por realizar llamadas externas al programa para poder realizar los entrenamientos y las clasificaciones, usando ejecutables compilados en C de la librería libSVM.

Esta librería la escogí debido a su gran popularidad en el mundo del machine learning, y es de código abierto, por lo cual no hay inconvenientes para usar su código bajo nuestro proyecto. La licencia de éste es BSD, por lo que tenemos libertad de usar éste tanto en código cerrado como abierto.

Mientras fui implementando el código para el módulo, pudimos ver entre los dos miembros que hay diferentes conexiones entre cada módulo, por lo que he estado colaborando con mis compañeros en el resto de módulos en lo que respecta al uso de ficheros de entrenamiento y en el entrenamiento y testeo en el módulo 3.

Entre todos también fuimos haciendo pruebas usando todos los módulos, tanto con imágenes de la cámara térmica de nuestro tutor, como con imágenes que pidió Alejandro para el estudio e identificación de la neumonía sobre pacientes. Entre todos pudimos unir ambos conocimientos para conseguir clasificaciones óptimas.

Por otra parte, también he ido documentando en la memoria todas mis investigaciones en el capítulo 2 y su funcionamiento en el módulo 2 sobre el capítulo 4 y sobre el apéndice B. El apéndice A también ha sido escrito por mí.

7.2. Alejandro Rodríguez Chacón

En el inicio del proyecto, en la fase de investigación, me encargué de familiarizarme con el estado del arte y las técnicas de procesamiento de imágenes, es decir, de la matriz de co-ocurrencia y de las fórmulas matemáticas que utilizamos para obtener las componentes del vector de características. Este estudio consistió en la lectura de la bibliografía aportada por el tutor(entre la que se encuentran [12] y [9]) y de documentos obtenidos de otras fuentes, como "Textural features for image classication"de Robert Haralick.

Por tanto, cuando comenzamos a desarrollar el sistema, mi cometido fue programar en Python aquellas partes que tenían que ver con esta investigación y posteriormente las conecté con la GUI. En definitiva, elaboré el módulo 1. Para poder comenzar a desarrollar tuve antes que investigar el funcionamiento de Python en algunos aspectos, ya que no tenía una gran experiencia previa con este lenguaje.

En este punto, aunque no de cara a la implementación del sistema, también comencé a estudiar otros aspectos importantes de la teoría como: SVM, los distintos modelos existentes para el procesamiento de imágenes digitales(estadísticos, estructurales, etc) y estudios de otros autores sobre la textura.

Durante este tiempo, también ayudé a resolver algunos problemas del módulo 3, sobre todo aquellos que tenían que ver con mantener la coherencia con el módulo 1.

Al obtener una primera versión de la aplicación, tras revisar lo hecho por todo el equipo, comprobé el comportamiento de los módulos y pude encontrar distintos errores como que el módulo de toma de muestras cogía el color de la marca(verde o rojo) como muestra o que las muestras se solapaban.

Una vez terminada la parte técnica del proyecto, también me he responsabilizado de hacer la mayor parte de pruebas iniciales para comprobar la efectividad de la máquina con problemas reales.

Estas pruebas conllevaron, con el fin de cubrir la mayor parte del abanico de posibles configuraciones, una planificación de las distintas combinaciones de parámetros que eran interesantes para probar en el sistema. Cuando finalizamos estos test, a través del análisis de los ficheros de salida del módulo 2, conseguimos definir que parámetros eran los más adecuados para problema propuesto.

En cuanto a la tarea de obtención de datasets para la investigación, mi contribución ha sido aportar el conjunto de datos de la neumonía y contactar, de manera infructuosa, con la universidad de universidad politécnica de Orleans (Francia) con el propósito de obtener un dataset de osteoporosis.

Sin embargo, diría que la mayor parte de mi trabajo ha estado destinado, en primera instancia, a la investigación de las tecnologías utilizadas y de las posibles aplicaciones del proyecto y, en último término, a la elaboración de la mayor parte de esta memoria, para lo cual ha sido necesario aprender LaTex desde cero pues ningún miembro del equipo tenía experiencia previa con este

sistema de composición de textos.

Capítulo 8

Apéndice A: Ejemplo de ejecución

Como ejemplo de ejecución tomaremos una imagen de neumonía bacteriana:

Figura 8.1

Usaremos la aplicación del módulo 1 para ir dibujando algunas muestras de neumonía y del resto del cuerpo. En total se toman 260 muestras: 130 positivas y 130 negativas:

Figura 8.2

Se ha decidido usar un tamaño de muestra 16, con distancia 1 y escalado de grises 16. Se han usado todas las características y todas las direcciones de la matriz de co-ocurrencia. Una vez hecho esto se pulsa sobre "Start"para procesar todas las muestras y generar el archivo zip con todos los ejemplos a entrenar y las configuraciones a usar en el módulo 3 dentro del archivo json.

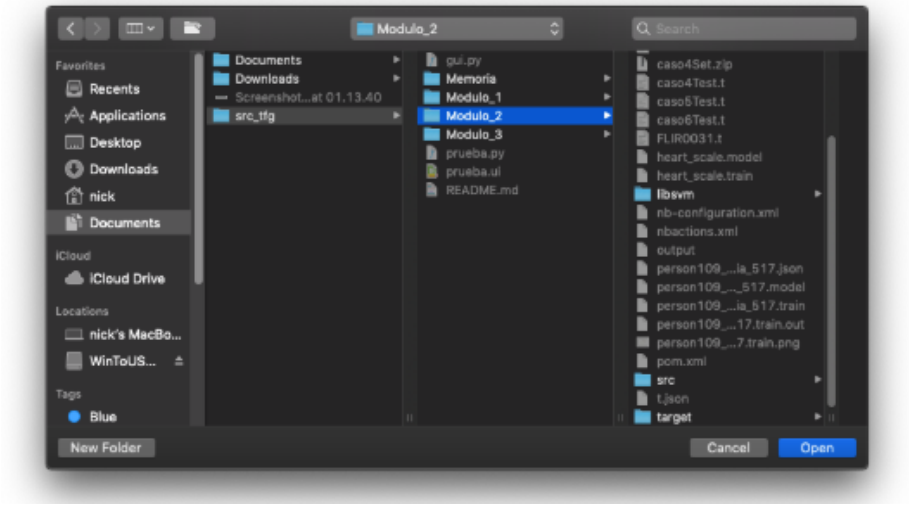

Figura 8.3

El archivo generado en este ejemplo se llama TrainingSet.

Ahora, para sacar la clasificación por SVM, se ejecutaría la aplicación del Módulo 2. Se carga el zip recibido del módulo 1 a la interfaz gráfica:

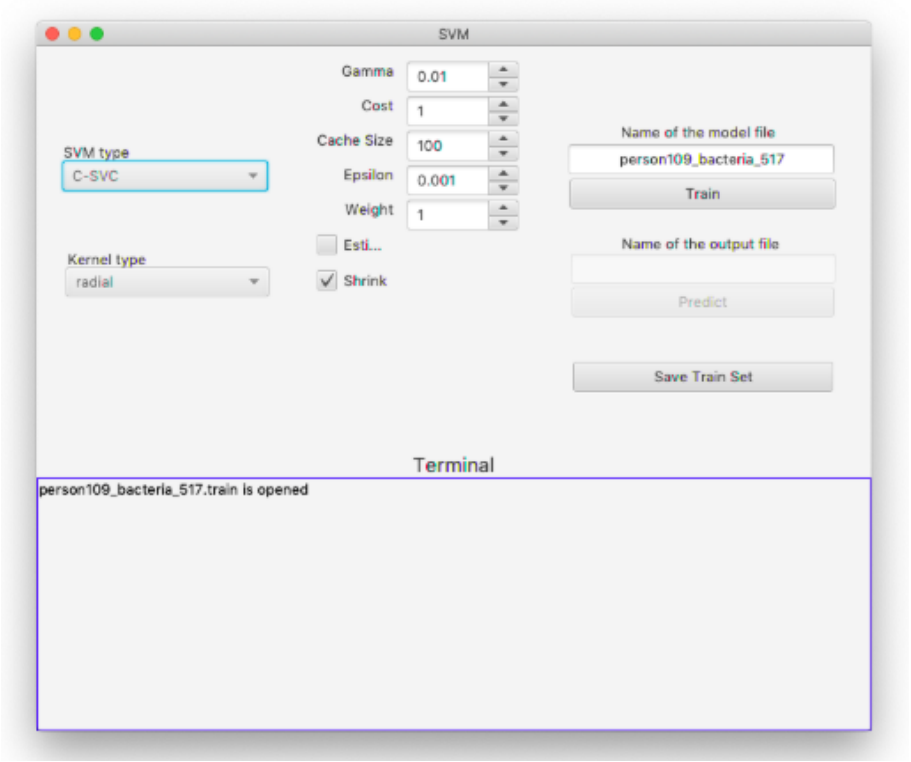

Figura 8.4

Vemos desde la parte de Terminal que efectivamente se abrió el dataset de entrenamiento en la aplicación.

Ahora se tocará el script python grid.py, con este podremos sacar parámetros C y gamma óptimos para clasificar en este caso sobre el kernel radial. En este caso, se escribiría sobre una terminal del sistema:
python libsvm/tools/grid.py
-svmtrain /Users/nick/Documents/src_ tfg/Modulo_ 2/libsvm/mac/svm-train
-gnuplot /usr/local/Cellar/gnuplot/5.2.6_ 1/bin/gnuplot /Users/nick/Documents/src_ tfg/Modulo_ 2/person109_ bacteria_ 517.train.

Las opciones -svmtrain y -gnuplot sirven para localizar los ejecutables del svm-train y del gnuplot para crear la gráfica. En este ejemplo se usan parámetros C y gamma que grid.py usa por defecto, para modificarlo habría que emplear las opciones -log2c y -log2g para definir los parámetros que queramos testear en la cross validation.

Una vez ejecutado, nos salen los mejores parámetros de c y gamma en cada iteración:

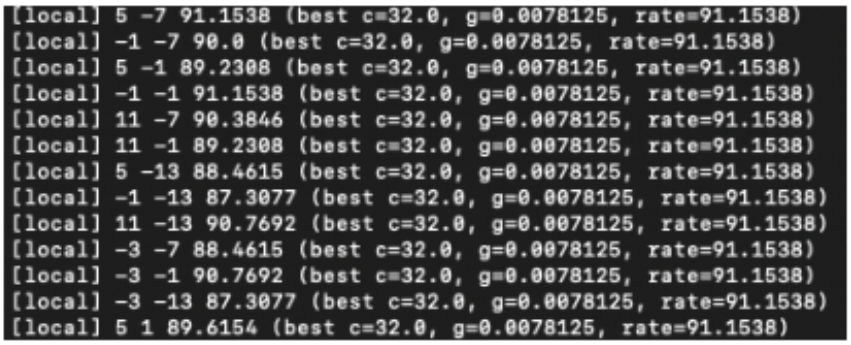

Figura 8.5

Vemos que en estas iteraciones los parámetros consultados sacan una precisión de 91.1538. Más adelante veremos que cuando decrementa C en las iteraciones, puede llegar a conseguir una precisión de 91.5385:

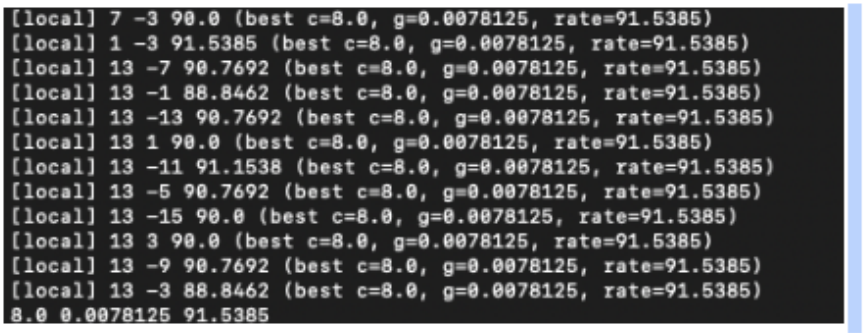

Figura 8.6

Los últimos valores generados son en este orden: coste optimizado, gamma optimizado y la mejor precisión con estos dos parámetros. Aparte de esta salida, veremos que en la carpeta donde se ha lanzado el comando se encuentra un archivo png con la gráfica:

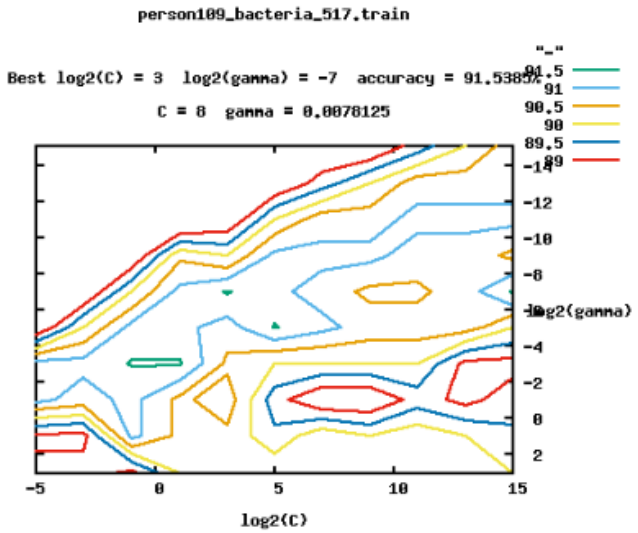

Figura 8.7

Las diferentes líneas representan las diferentes precisiones obtenidas por los valores de gamma y C representados sobre el eje cartesiano.Con estos datos sabemos que el mejor C para usar en este dataset es 8 y el mejor gamma 0.0078125. De este dataset sacaremos 87 ejemplos al azar para emplearlos en un nuevo archivo de test.

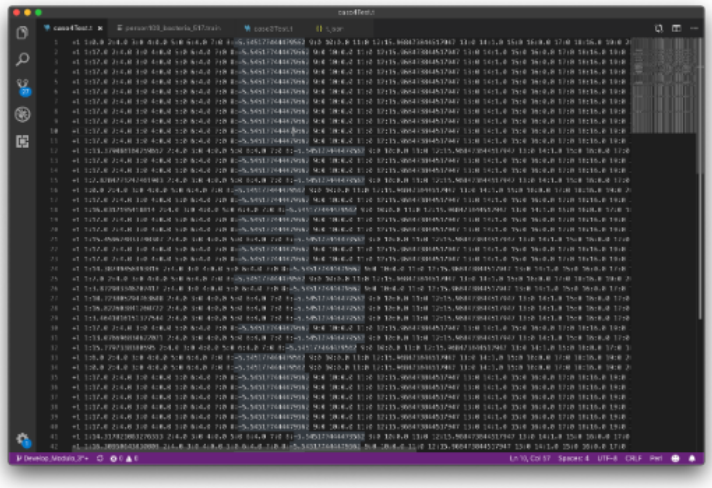

Figura 8.8

Ahora con el archivo train sin esos casos de test, realizaremos el entrenamiento sobre él:

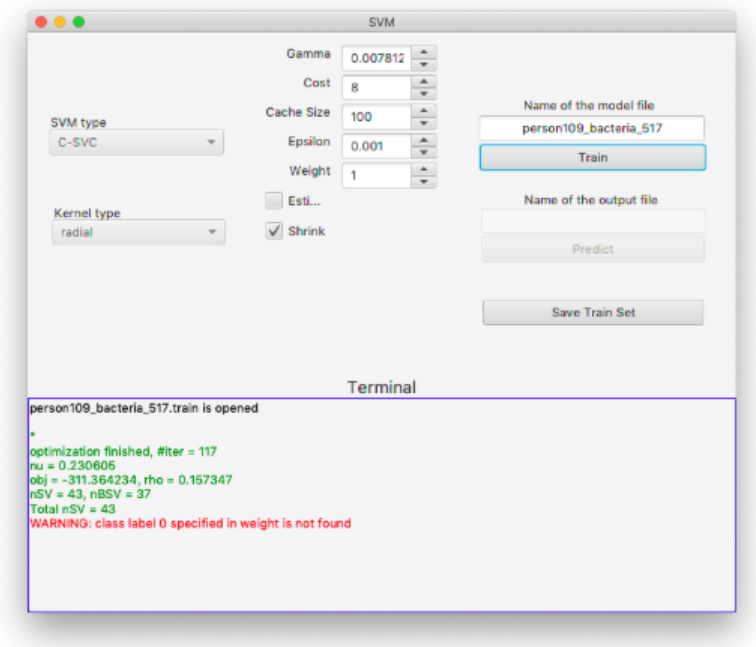

Figura 8.9

Ahora que tenemos el archivo model para poder clasificar, introducimos nuestro archivo de test y predecimos los resultados de éste:

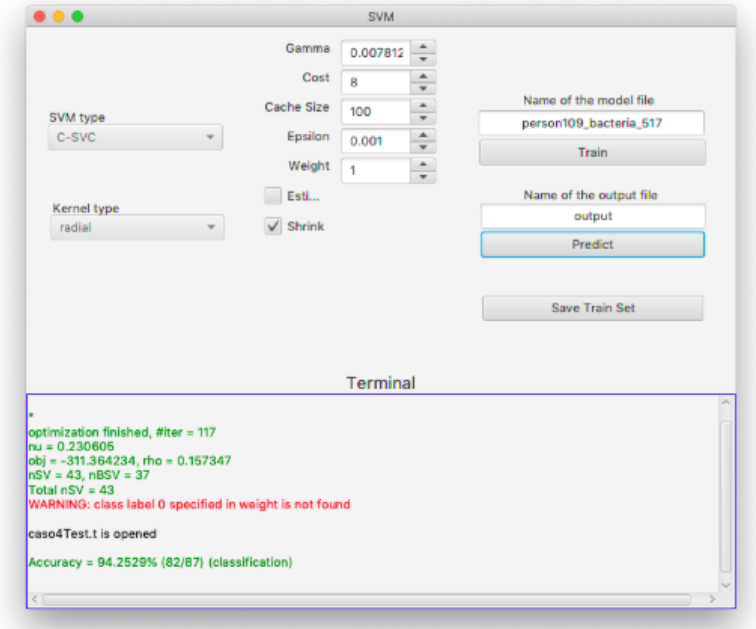

Figura 8.10

Por lo que podemos ver, esta configuración de C y gamma ha resultado muy buena para la máquina, un 94.25 % de aciertos. Podremos guardar un nuevo zip con el comando usado en el entrenamiento usando el botón "Save Train Set". Este comando en concreto se guardará sobre el archivo de configuración json que hemos recibido por parte del módulo 1 y será de utilidad de cara al módulo 3.

Una vez que tenemos todo el zip preparado, procedemos a emplear la interfaz del módulo 3. Cargamos un set de imágenes que están en un archivo zip pulsando en Cargar imágenes. Nos saldrá una ventana para escoger el set de imágenes:

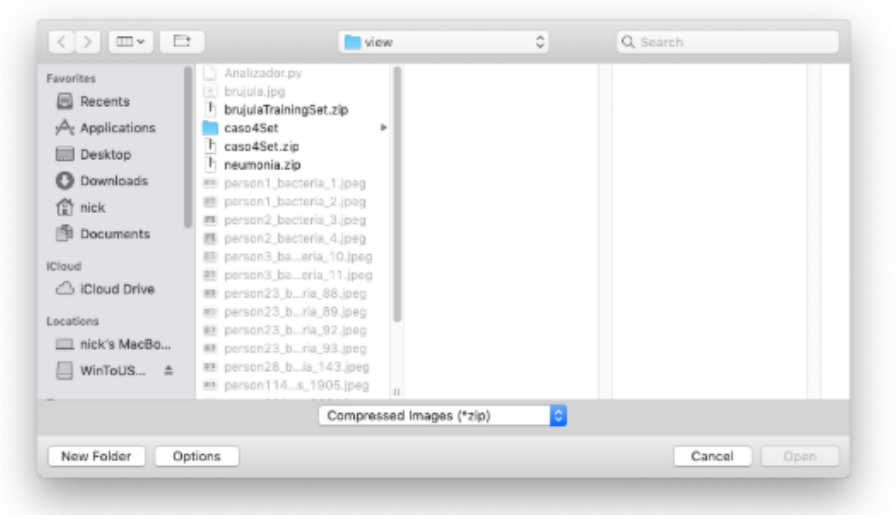

Figura 8.11

Sólo puede aceptar comprimidos en formato zip. Una vez cargado el set de imágenes cargaremos nuestro zip de training, que ha sido modificado tanto en el módulo 1 como en el 2, usando el botón Cargar Training Set:

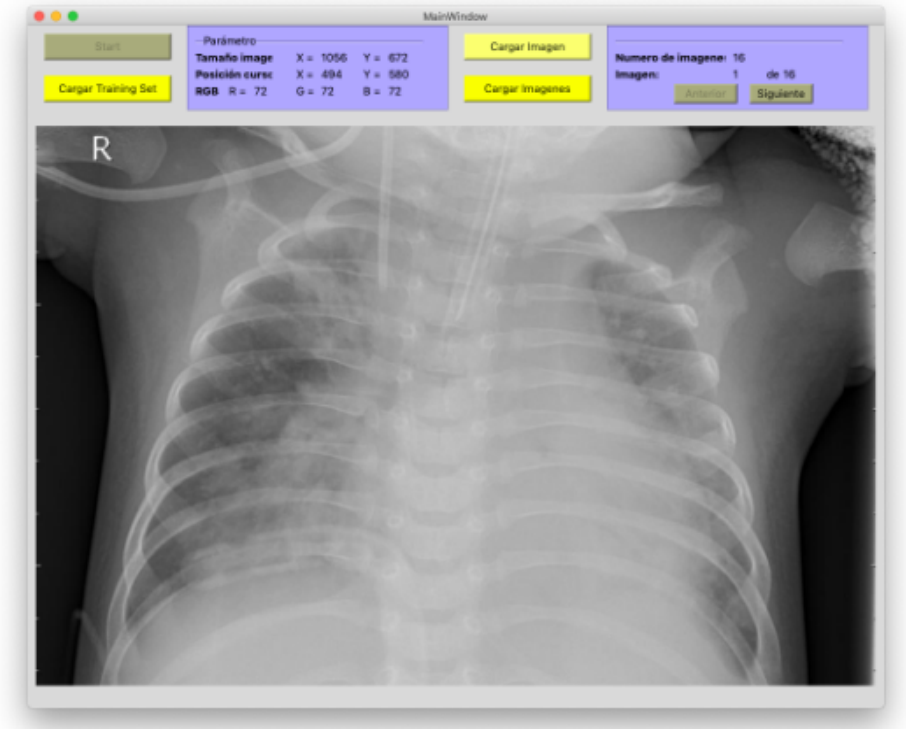

Figura 8.12

Esto nos abrirá otra ventana para escoger nuestro zip. Esta ventana solamente admitirá formato zip:

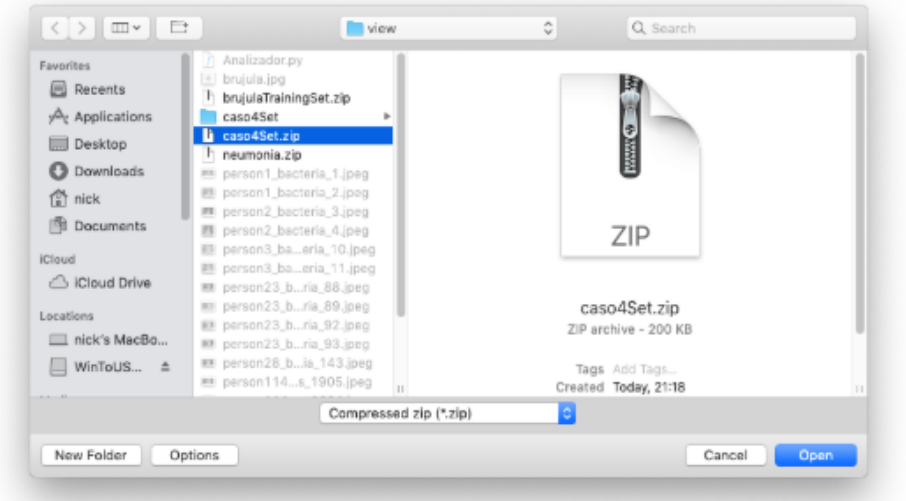

Figura 8.13

Ahora solo nos queda apretar en "Start"para empezar con el barrido de imagen y generar todos los casos de neumonía que encuentre el SVM.

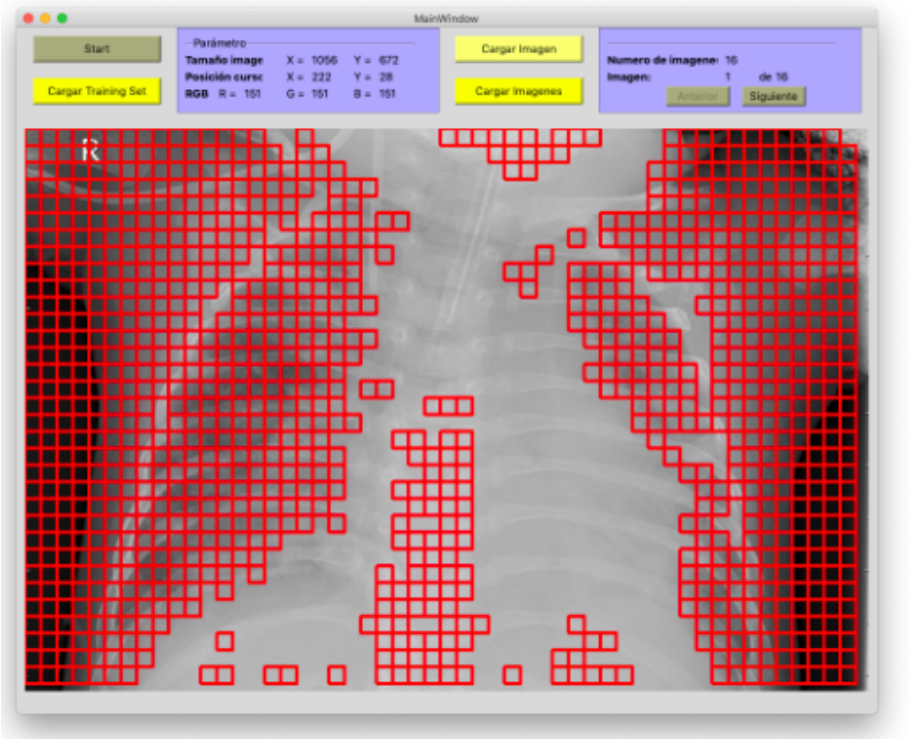

Figura 8.14

Capítulo 9

Apéndice B: Manual de usuario

Para empezar, lo primero es obtener los diferentes módulos, para el primero y el tercero requerimos Python 3.7 y ciertos módulos instalados, por el otro lado el módulo 2 requerirá OpenJDK 12 y ya pues al estar montado con maven el mismo cargará todas sus dependencias.

Para instalar las dependencias de Python una forma recomendada es hacer uso de anaconda pues se adapta a los diferentes entornos.

Una vez instalado desde la consola de anaconda (conda prompt) basta ejecutar los comandos conda install:
conda install -c anaconda numpy
conda install -c anaconda pil
conda install -c anaconda scipy
conda install -c dsdale24 pyqt5

El resto a excepción de Tools debería venir instalado por defecto, con la instalación de Python 3.7 de anaconda

Aquí resumiré las tecnologías y herramientas necesarias para la ejecución completa de nuestra herramienta.

Requisitos:

- Clasificador (Módulo 1)
- SVM (Módulo 2)
- Analizador (Módulo 3)
- Python 3.7
- Numpy

- Pillow
- Scipy
- PyQt
- Tools (Python lib)
- ZipFile (Python lib)
- Math(Python lib)
- IO (Python lib)
- OpenJDK 12

Bibliografía

[1] V. Arguedas. *La geometría de la naturaleza: Benoit Mandelbrot.* Escula matemática, Universidad de Costa Rica, Febrero, 2012.

[2] A.Rosenfeld and E.Troy. *Visual texture analysis.* 1970.

[3] V. Bush. *As we may think.* The Atlantic Montly, Julio, 1945.

[4] M. M. Galloway. *Texture analysis using gray level run-lengths.* Computer Graphics and Image Processing, 1975.

[5] S. K. D. Haralick, R.M. Textural features for image classification, 1973.

[6] H.Kaizer. *A quantification of textures on aerial photographs.* Boston Univ., 1955.

[7] A. Kowalczyk. *Support Vector Machines Succinctly.* Syncfusion, 2501 Aerial Center Parkway Suite 200 Morrisville, NC 27560 USA, 2017.

[8] F. J. M. Navas. *Desarrollo de un proceso de deconvolución para imágenes usando GPUs.* Facultad de informática, Universidad de Murcia, 2009.

[9] M. Presutti. *la matriz de co-ocurrencia en la clasificación multiespectral: tutorial para la enseñanza de medidas texturales en cursos de grado universitario.* Universidad Nacional de La Plata, Agosto, 2004.

[10] V. J. R.Bixby, G. Elerding and R.Loewe. *Natural image computer.* 1967.

[11] J. A. Rodrigo. *Máquinas de Vector Soporte (Support Vector Machines, SVMs).* https://rpubs.com/Joaquin_AR/267926, Abril, 2017.

[12] J. L. G. Rodríguez. *Estado Actual de la Representación y Análisis de Textura en Imágenes.* Julio, 2008.

[13] X. Wang et al. *ChestX-ray8: Hospital-scale Chest X-ray Database and Benchmarks on Weakly-Supervised Classification and Localization of Common Thorax Diseases.* National Institutes of Health, Bethesda, MD 20892, National Center for Biotechnology Information, National Library of Medicine, Department of Radiology and Imaging Sciences, Clinical Center.